Product Family Design Using Product Simulation and Multi-objective Optimization

Thesis

Product family design using product simulation and multi-objective optimization

Miguel Zapico

The Open University

STEM

Department of Engineering and Innovation

A thesis submitted in accordance with the requirements for the degree of Doctor of Philosophy

Declaration

This thesis is the result of my own research and does not include the outcome of collaborative work, except where stated otherwise. The dissertation has not been submitted in whole or in part for consideration for any other degree.

Miguel Zapico

Department of Engineering and Innovation

STEM

Open University

September 2019

Abstract

This study is concerned with the design of a range of products intended to cover different applications. The prominent example throughout the thesis is that of a family of industrial trucks that need to cater for a wide range of load capacities. That product range is normally built around platforms, i.e. basic sets of components that are common to some or all of the products in the range. With this approach, each product is made up of those common components and additionally other components that are specifically suited for each particular product.

The outcome of this thesis is a novel method to assess the possible combinations of common/specific components to build up a product range to cover a predetermined set of user applications and provide the company with a clear view of the trade-off between offering customer appealing products and keeping the costs down.

The method uses a combination of mathematical modelling and simulation for estimating the relevant performance attributes of each possible product design, fuzzy logic to reduce the naturally large number of objectives to a manageable one and a multi-objective searching algorithm to find a Pareto set of solutions to provide the decision makers with clear and useful information with which they can take a better decision.

Acknowledgements

This thesis is the result of 6 years of admittedly quite demanding work on top of a full time job. Throughout that time, there have been several people who have helped me one way or another, and I'd like to take this opportunity now to say thank you to them.

First of all, my supervisors Chris, Claudia and Iestyn -in alphabetic order- for trusting me, all their thorough and amazingly fast feedback, guidance and inspiration. They have been a fantastic team of supervisors that made me look forward to each time that I took the trip to Milton Keynes or they came my way. Always a pleasure to talk to them.

I would like to dedicate this thesis to Rebeca, Luis, Pablo, Alex and Gwen, aka my wife and kids. They have shown large amounts of patience and love coping with a stressed grumpy man writing his thesis.

Mum and Dad, there are many stages to go through in life before even starting a PhD, and I would never have come anywhere closer to that point without their support.

I'd also like to say thank you to all the people in the Engineering Concept Centre of Hyster-Yale, and in particular to Nick and Mark for their support for this project. I have enjoyed a lot my time there and that will always be kept in my memory as a very special period of my life.

Who says that water must always flow down to the earth? What humans call laws of nature are nothing but a generalization of observed events that they can not fully understand

Dohko, Gold Saint of Libra

I find your lack of faith disturbing

Darth Vader

Contents

List of Figures

List of Acronyms

2FFL	2 stages Full Free Lift
3FFL	3 stages Full Free Lift
ANSI	American National Standards Institute
CAD	Computer Aided Design
CCTV	Close Circuit Television
CE	European Approval Marking
CEN	European Committee for Standardization (Comite Europene de Normalisation)
CFD	Computational Fluid Dynamics
CI	Commonality Index
DFMEA	Design Failure Mode and Effects Analysis
DoD	Department of Defense
DRM	Design Research Methodology
FEA	Finite Element Analysis
ICE	Internal Combustion Engine
ISO	International Standardization Organization
ISS	Integrated Side Shift
KPI	Key Performance Indicator
LFL	Limited Free Lift
LPG	Liquefied Petroleum Gas
MOPSO	Multi Objective Particle Swarm Optimization

ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PFD	Product Family Design
PPD	Product Platform Design
QFD	Quality Function Deployment
R&D	Research and Development
TCO	Total Cost of Ownership
VDD	Value Driven Design
VDI	Association of German Engineers (Verein Deutscher Ingenieure)
VNA	Very Narrow Aisle

Chapter 1

Introduction

In the automotive industry the desire of companies to increase their offer and simplify maintenance and spare parts has led to the introduction of product platforms. Now many other application areas and sectors are following suit.

This thesis addresses the design of a range of products intended to cover different applications and the definition of the platform/variants structure. The *platform* is the set of common components and the *variants* the different products that compose the range. The aim is to present a method to decide what parts can be made common to all or some of those products and what parts should be kept as individual to each product variant. Several methods exist for solving this type of problems but most of them either start with a preconception of what parts will be shared, or restrict the problem by choosing the common parts first without the benefit of the big picture, or define a target that misses important points and trade-offs. This thesis will argue that these aspects are important for the industry and present a novel method with that in mind. The thesis will use industrial trucks as a prominent example of an industry immersed in the design of product families. The method will be developed, tested and validated using those products as a case study.

1.1 Background

In the early 20th Century, Ford started mass-producing their model T in what has become one of the most re-told episodes in every account of the history of manufacturing. All the cars coming out of the production line were built to the same specifications, to the extent that they were all painted the same colour. Ford sold millions of virtually identical units of the model T to customers of all kinds. By the 1960's, the American automotive market had significantly evolved. Each car producer offered many models and each model was updated every year. Customers were also offered a wide range of options from which they could choose the final specification for their purchase. This made it theoretically possible for a company to manufacture millions of cars without repeating the same exact configuration twice (Wilson, 1997). The automotive industry was a pioneer in this approach which, since the 1990s, has been widely known as mass customization (Da Silveira et al., 2001). Nowadays, it is often expected that companies producing consumer goods are flexible enough to offer a range of products that appeal to a wide customer base, by satisfying the particular requirements of different kinds of users, without severely increasing the costs. For companies that produce goods destined for industrial use, personal preferences are arguably less important because the products are mainly perceived as tools or components. But, the applications for which the products are going to be used can vary widely, and the implications for the manufacturer are still the same, i.e. they need to offer a range of products flexible enough to satisfy the customer's desires at a competitive price.

With this in mind, companies that design and manufacture goods aim to offer a wide variety of products while at the same time using the minimum number of different parts and systems (Nelson et al., 2001). This approach has several advantages: economies of scale and reduction of parts inventory (Martin and Ishii., 1996); shorter lead time and easiness to design new product variants (Meyer and Lehnerd, 1997)

(Ulrich, 1995); and a reduction of manufacturing tooling and processes (Sidique et al., 1998). Another indirect advantage is that a well thought out product platform with appropriate margins for future change will lower the redesign costs in the long term and will delay the necessity for a completely new design (Eckert et al. 2012) (Lebjioui, 2018). The concept behind this approach is known as product platform design (PPD) and is central to this thesis.

Following that approach, the products, known as product variants, that a company offers are usually organised into product families, i.e. *sets of products that share a number of common components and functions with each product having its unique specifications to meet demands of certain customers* (Pirmoradi et al., 2014), and hence it can be said that product families are built around product platforms, with the family being the set of products that the customer can buy and the platform the structure of common components and principles.

Figure 1.1 shows a family of industrial trucks, the type of product that will serve as a case study throughout this thesis. These trucks are a good example of products that have similar functionality, lifting and moving loads. However, they are also vastly different depending on the application for which they are used, ranging enormously in terms of size and power. This leads to a non obvious question of what components can be common to what products without subsequent underperformance or over-design issues. The picture is a good visual clue of the range of applications and the different sizes available; a van is placed in the middle for reference. The size of the biggest of them, the reachstacker, can be inferred from the fact that only a small part of it lies within the picture frame.

Figure 1.1: Materials handling trucks from smallest (hand pallet truck) to largest (reachstacker)

1.2 Motivation

The topic of this thesis is how a company, or a design team, can conceive an offer of related but different products for different applications while being competitive both at the market place and also with the costs involved in designing and producing those products. The design team also has to take into account the present situation in which the company may have existing individual products or components.

The main idea is to provide information about the potential strategies for sharing components across those products to keep both aspects within target.

This is a well-known problem in the industry, and one which the author is familiar with after being an R&D engineer for several years at the time of conducting this study. During that time, the author has been exposed to the realities and difficulties of designing products and product families, and seen how many decisions are regularly taken based on history, preconceptions or judgements without the benefit

of having quantitative data at hand. For example, it is common to have a project to develop a lifting mechanism for forklift trucks with capacities between 2 and 3.5 tonnes, while at the same time there is another project to design a similar lifting mechanism for trucks between 4 and 7 tonnes. Why two different mechanisms? and why the dividing line is between 3.5 and 4 tonnes? What if only one mechanism were designed? or two but with a different point marking the division?

This kind of decisions are taken at the initial stages of a design effort and have a profound effect downstream, imposing restrictions on the detail stages that will provide the final output. In particular, the class of decisions that will be discussed in this thesis are those related to the structure of a product line, what and how many products will be on offer? what is the relation between them? what components will be shared across what products? and what components will be made individually for a particular product?

This thesis draws heavily from the particular industry of material handling vehicles, or industrial trucks. Those vehicles all share a common purpose to move loads from one place to another, but there are still remarkable differences, the most obvious one is the size of the loads, which can range from small boxes in a store to large containers in the docks. At the same time, industrial vehicles are complex machines and their fitness for purpose is measured against a varied set of attributes, some of them in conflict to each other. This means that it is hard to define an appropriate criterion to say that a particular vehicle is better or more attractive than other. Even if that could be defined, it will not be trivial how to improve a product, due to the complex relation between components and how they affect the different attributes. Additionally, in this kind of industry it is not common to design a product offer from scratch, models tend to be introduced sequentially, and new models are sometimes influenced by existing models and components that are already in use. All this makes this industry a compelling case for study.

1.3 Objectives of this thesis

The objective of this thesis is to propose a method to design the platform architecture, or the structure of common components across a product line intended to cover a number of applications. The aim is to avoid any preconception of how many products are necessary for those applications and assess the potential solutions against a multi-attribute criterion that considers both the performance of each product, how fit they are for each application, and the costs and savings that can be achieved by sharing components across different products. The method will be demonstrated on an industrial case study that will be introduced in chapter 4 and described in detail in chapters 6, 7 and 8.

Figure 1.2 shows an example simplified for the purpose of clarity. The starting point is the top and bottom lines: 6 applications and a number of choices for engines, pumps, cylinders, gearboxes and tyres. The result of solving the problem is the map between the top and bottom lines, and consists on 4 different products build from the combinations of components indicated by the arrows.

The objective of this thesis is to develop an algorithm that allows the user to generate solutions of that type and evaluate them according to a multi-objective criterion to provide a reduced set of solutions showing the trade-offs over which the final decision needs to be based.

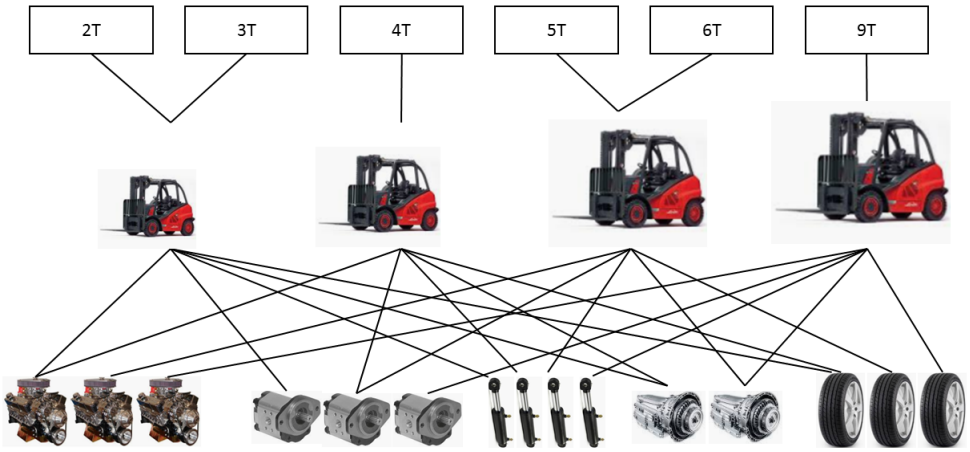


Figure 1.2: Example of applications to products map

1.4 Research questions

There is a set of research questions for which this thesis will attempt to find an answer. Those questions are:

Research question 1

How can a platform strategy improve product development processes?

Research question 2

What are the barriers to industrial adoption of product platform strategies?

Research question 3

Can an alternative product platform strategy be devised that addresses these barriers?

Research question 4

How well could this new strategy perform in a real world industrial context?

The nature of question 1 will be justified in the literature review, where the main characteristics of existing methods are explained as well as their shortcomings due to some a priori decisions that are commonly taken and restrict the potential solutions to the problem.

The second question addresses an important issue of complexity associated to this problem, and will be clearly reflected in the case study, i.e. the vastness of the design space, the need for simultaneously assess the different products that compose a family considering the multitude of performance attributes that are relevant, and the presentation of the solutions in a meaningful and useful way. As it will be seen in the literature review, the more attributes that are considered, the harder it becomes to distinguish between desirable and non-desirable solutions. This inevitable fact leads to potential scenarios in which showing the solutions to a problem may obfuscate the field to the point of not being helpful to the person who makes the final decision.

The third and fourth questions incorporate typical characteristics of real industry problems that complicate the process to solve them and are often ignored, such as the diverse nature of the variables under consideration and the fact that there is often a legacy element in most new designs. The method presented in this thesis attempts to be general, and capable of dealing with the different scenarios that can be found in industry settings.

1.5 Thesis structure

After this introduction, this thesis will be structured in the following chapters:

- Chapter 2 - Literature review. A review of the state of the art in the field and related fields relevant to this project.
- Chapter 3 - Methodology. Description of the methodology followed during this project, and the frame in which it is placed.
- Chapter 4 - Industry justification. Description of the problem from the industry point of view, where the motivation for this project comes from. This chapter is written based on the author's experience.
- Chapter 5 - Developing a method for product family design. A general description of the philosophy and the method proposed in this thesis.
- Chapter 6 - Case study. An application of the method described in the previous chapter going step by step on a real case study.
- Chapter 7 - Implementation and results. Implementation of the case study and presentation of the results.
- Chapter 8 - Validation. Validation of the results presented in the previous chapter.
- Chapter 9 - Evaluation, discussion and conclusions. A review of the thesis and how the method addresses the original research questions and makes and original contribution. Final thoughts and paths for further research.

Figure 1.3 shows a map between the research questions and the chapters of the thesis in which they are addressed.

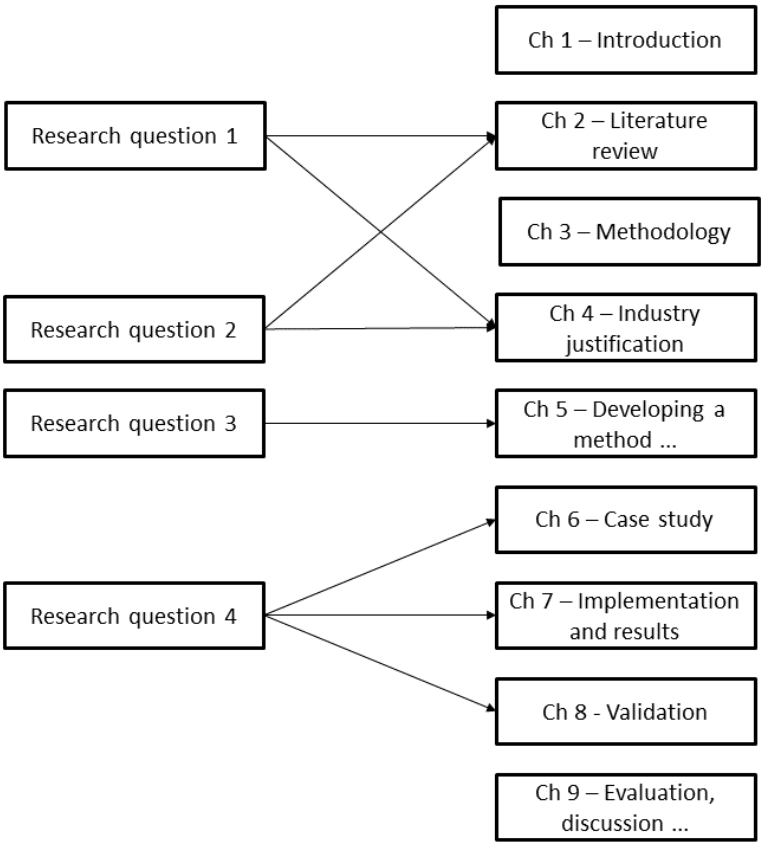


Figure 1.3: Map between research questions and thesis chapters where they are addressed

Chapter 2

Literature review

The research questions 1 and 2 refer to how a platform strategy can improve product development and the barriers to industrial adoption. In this context, development not only refers to the process of development itself but also includes the outcomes and how desirable they are for the company. This chapter is a review of the existing literature in that field to highlight the pros and cons of platform strategies as well as different methods, approaches and debates around the issue from an academic point of view. To answer those research questions, the analysis will be completed in chapter 4 from the industrial point of view.

The structure of this chapter is shown graphically in figure 2.1 and it is divided in two distinct parts: The first part is a review of the state of the art regarding product family design, it begins with a description of product architecture, which is a fundamental concept to understand the relation between the applications, products and components in a product line-up and its organization in platforms and variants. This is followed by an introduction to platform design, with definitions, pros and cons, types, typical design process and existing methods. This part ends with the identification and statement of an existing gap in current research that led to the formulation of the research questions and a high level introduction to the method that will be proposed and discussed in the remainder of this thesis.

The second part is a general background on fields and techniques that the novel method will draw upon. Being this thesis focused on complex products, some kind of criterion has to be defined to balance the different targets both for each product and the entire product range, leading to an introduction to decision making theory and the review of two different approaches representative of the two main currents of thought: value driven design (an attempt to find a single measure that defines a whole product) and multi-objective optimization, where each target is treated separately and the study focuses on how each individual target is met. The latter will lead to the method proposed in this thesis in combination with a fuzzy method to compare physical performance of different attributes for each product, which is why some basic notions of fuzzy mathematics are also included in this chapter. The final field to be reviewed is that of modelling and simulation, which is necessary to estimate the required characteristics of the potential products without building any prototype.

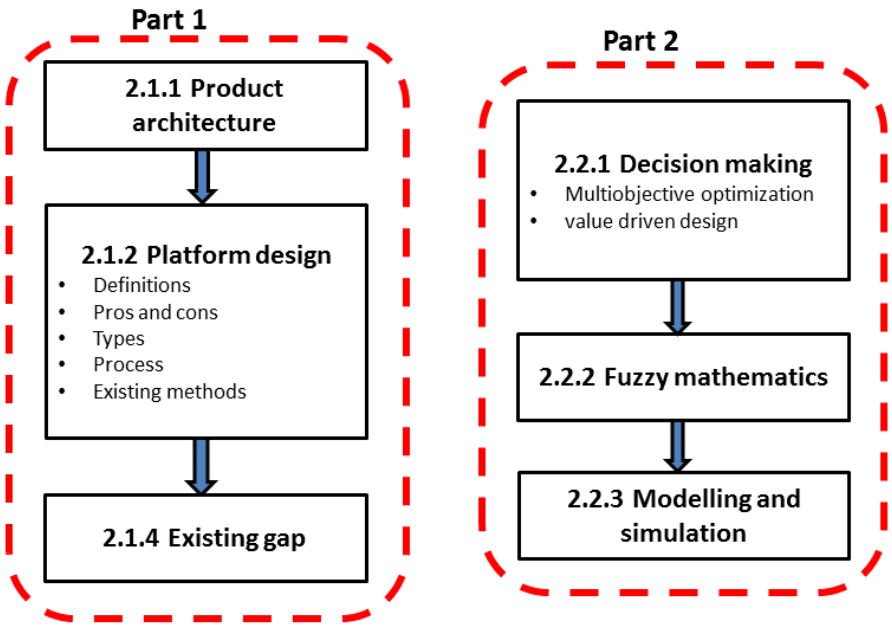


Figure 2.1: Literature review scheme

2.1 Part 1: Product family design

This part will set the context and show the existing research around product family design focusing on the most important aspects for this thesis. Product family and platform design has been a research topic since the 1990's and it is still a very active issue both from the academic and industrial point of view.

2.1.1 Product architecture

A fundamental concept for this project is that of *product architecture*. Eppinger and Ulrich (2012) define the product architecture as *"the scheme by which the functional elements of the product are arranged into physical chunks and by which the chunks interact"*. Products can be based on a modular architecture if the *"chunks imple-*

ment one or a few functional elements in their entirety and the interactions between chunks are well defined and generally fundamental to the primary functions of the product" (Eppinger and Ulrich, 2012), or integral where either a chunk implements several functions, or a function is performed by several chunks, or the interactions between them are not well defined. Real world complex products are not necessarily based on purely modular or integral architectures, but a combination with a certain degree of modularity. Highly modular architectures are more suitable for designing product families with a high level of commonality and flexibility, whereas integral architectures may be the preferred option when commonality is not so important and technical constraints are the main issue (Pirmoradi et al., 2014).

A product family can be defined as a set of product variants based on a common platform but otherwise different to satisfy varied customers (Nayak et al, 2010) or in other words, a set of products sharing some common components and functions while still catering for different customer demands (Meyer and Lehnerd, 1997). The problem of designing one or more product families is referred to as *product family design*, and normally abbreviated as PFD. Family architecture is a term that considers how the different products of a family interact with each other in terms of what commonality exist among them, as well as the modularity of each product (Jiao, 2000). Figure 2.2 shows an example of a product family composed of four product variants. Components A and B are common to all the variants and hence they are the common platform, whereas components D and E are individual for each product. Components C1 and C2 are common to some variants, this concept is referred to as subplatform and will be explained later on. That commonality among different products leads to the related field of platform design.

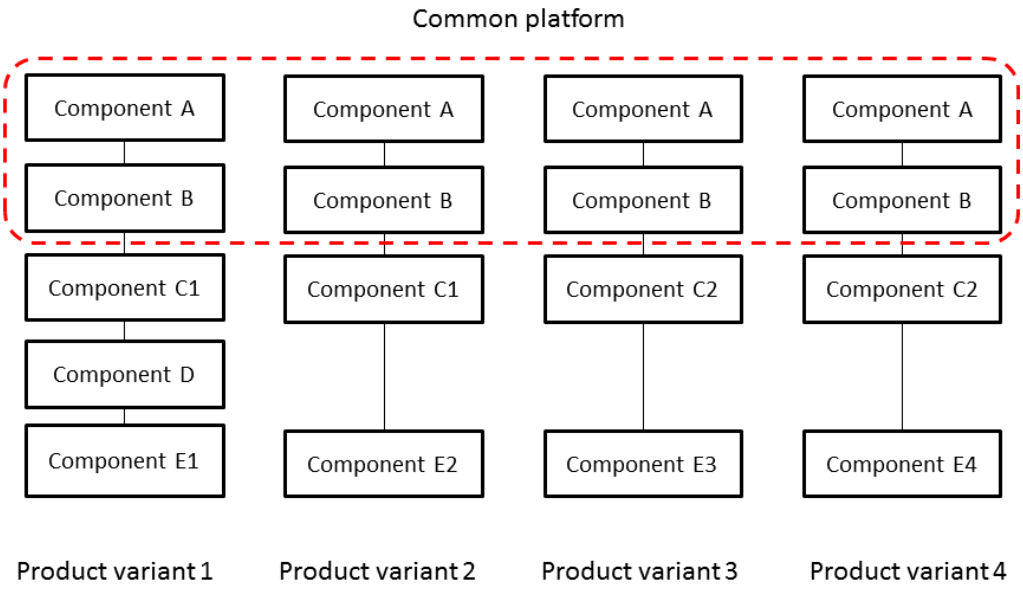


Figure 2.2: Example of product family composed of four product variants

2.1.2 Platform design

When a company wants to offer a range of products for different applications, known as *product variants*, the simplest way to have the products with the best possible performance is to optimize them individually. This will generally result in a collection of products completely different to each other, and this approach is known as a *null platform* (Nelson et al, 2001). However, if the number of different products is big, with big meaning more than 2 or 3, as can be expected in many sectors nowadays; this approach may lead to prohibitively high costs as the company would miss on the benefits of economies of scale associated to the sharing of some common components, both in terms of procurement, development and maintenance, and the products may not be competitive in the market against other products that may be inferior in terms of performance but significantly cheaper to purchase, making them

more appealing to the customer overall (Meyer and Lehnerd, 1997).

A common way to offer a wide range of different products while at the same time keeping the costs under control is what is known as *platform family design*. Although it is difficult to find a widely accepted definition, the idea is to design different product variants based on both common components across the range and also specific components for each product. Typical examples are different automobiles sharing parts such as the chassis, engine, suspension and transmission, while each model has unique body panels, windows or interiors. Another common example is domestic appliances such as dishwashers or washing machines sharing the main body frame and electronics but different mechanisms inside.

The term *commonality* refers to the fact that the same parts or systems are used in the manufacture of different products (Cambridge dictionary, 2019). Those parts or systems are known as *components*, and the number of them in which a product is divided is not defined by a unique criterion, since a component can be an engine or a single bolt depending on the level of detail applied. This level of detail or granularity has a deep effect on how the problem is posed (Holtta-Otto et al., 2014) as the number of potential design combinations grows exponentially with the number of component choices and this number will be larger the finer the granularity (Maier, 2016). Components can be common or individual, depending on whether or not they are shared by two or more product variants or they are explicitly designed for just one variant. The practice of sharing components across several products is not a target in itself, it is a worthwhile pursuit only if the benefits associated in terms of increased revenue or cost reduction overcome the downfalls (Cameron and Crawley, 2014). The practice of sharing common components across different products without a well thought strategy may result in compromised designs that would not be as optimal as they could have been without any commonality restriction. Either each individual product fails to deliver the maximum possible performance (Nelson et al, 2001),

or conversely, products can be over-designed resulting in excessive running costs (Eckert et al., 2013). On the other hand, optimizing products individually results in very specialized and expensive products that are unlikely to be competitive in the market. For most types of products, the design of a product range involves a trade-off between the advantages of component commonality, primarily economic but recently also in terms of sustainability (Kin and Moon, 2017), and the level of performance of each individual product (Alizon et al., 2007) or in other words, between the economy of scope from product variety and the economy of scale from component sharing (Song et al., 2019). It is inherently a combinatorial problem with conflicting objectives (Eichstetter et al., 2015).

Definition of platform

The exact definition of the term varies among different authors and texts. For example, for Eppinger and Ulrich (2012) a product platform is *"the collection of assets, including component designs, shared by otherwise different products"*. Whereas for Simpson and D'Souza (2004) it is *"a group of related products that share common components and/or subsystems"*. Robertson and Ulrich (1998) go further and include the people involved in the development and their relationship, and Nayak et al (2002) define platform as *"the maximum level of standardization with which it is possible to meet all the requirements for all the products"*.

Although the definitions vary, the idea of what a platform strategy means for the design of a product range appears more or less clear and can be summarized as *"essentially an effective and deliberate program of component reuse which takes advantage of the economies of scale across the product family, while minimizing the negative impact of reuse on individual product variant distinctiveness and performance"* (De Weck and Suh, 2003). However, the idea of platform is not restricted to sharing physically equal components, it can also include scaled components that

share the same principles, as for example in different length trailers sharing the same manufacturing process, or even intangible assets such as software modules (Sangiovanni-Vicentelli and Martin, 2001).

As there are multiple definitions of platform available in the literature, it is necessary to adopt one that will be followed for the remainder of the thesis. The term *platform* will refer to the set of common components shared across different products of the family, in opposition to *individual components* which only appear in a particular product variant.

Advantages and disadvantages

The design of platform based product families comes with some advantages. Cameron and Crawley (2012) classified those advantages in three groups: revenue benefits, cost savings and risk benefits.

The revenue benefits are mostly due to the shorter lead time and easiness to design new product variants to fill a niche (Meyer and Lehnerd, 1997)(Ulrich, 1995) or satisfy growing customer demands (Aljorephani and ElMaraghy, 2016). This point is especially important for the type of products that are configured or engineered to order (Levandowsky et al., 2015), where a product platform must take into account a wide range of potential requirements and products not yet defined.

Among the cost savings are the use of economies of scale to get cheaper components, reduction of parts inventory (Martin and Ishii, 1996), and a reduction of manufacturing tooling and processes (Sidique et al, 1998). Pander (2012) and Cameron and Crawley (2014) estimated the benefits in cost reduction can be up to 30% and production times cut by half. The costs advantages are difficult to estimate, as they are several types of costs that depend on the family structure, such as development, administration, testing and parts. The parts costs are some of the most affected by the different degrees of commonality, and also one of the simplest

to estimate. Ripperda and Krause (2017) estimated a share at 40 - 50% of the family cost. Another advantage, especially for multinational corporations, is that a wide product platform allows for easier relocation of production to better suit the market needs (Lampon et al., 2017)

The third group, risk benefits, are associated with the use of known technologies and processes, the risks of failures in new products is obviously lower when parts or engineering principles of that product are already incorporated in other products and it is known how they perform. However, it is not all advantages in structuring products around a platform, there are also some disadvantages associated, such as:

- Cannibalization between the different variants (Kim and Chhajed, 2000)(De Weck and Shu, 2003), i.e. the possibility of a lower quality product affecting the sales of the higher end product due to them being perceived as similar and the customers not being willing to pay the extra premium for the higher end product. A typical example was the increase of the Skoda branded cars to the detriment of the more expensive Volkswagen when they started to share an important percentage of their parts (Shu, 2005).

- Performance of each individual product based on a platform can be poorer than it could be without any commonality constrain (Nelson et al, 2001). This makes platform strategies not optimal for very specialized products or those for which performance are almost the only important features, such as fighter jets or expensive supercars. It is, however, still possible to find examples of commonality and platforms even in products as unique as NASA spacecrafts (Gonzalez-Zugasti et al, 2000), although in those cases commonality is only accepted when performance is not affected.

- Flexibility limits. New advances and requirements may eventually make platforms obsolete and very expensive to replace. A company may try to over-

stretch a platform life due to the high costs of developing a new one, and this will result on the product features or performance being restricted. (Eppinger and Ulrich, 2012).

- Unexpected technical problems. An example was the Audi TT, whose rear end tended to be looser than expected. The problem was traced to the car being based on a platform designed for other different cars without taking into account the particularities of the TT (De Weck and Suh, 2003). For products with a high degree of complexity it can be very difficult to understand and predict all the consequences of using an existing basic design, although these problems also arise for individually designed products.

When designing a platform or a product range based on one or more platforms, engineers need to balance the trade-offs between these advantages and disadvantages in order to achieve the best range to meet the company objectives.

Types of platforms

Product platforms can be classified according to several criteria. In terms of how the platform is introduced and evolves, there are mainly two different and opposed approaches (Kalligeros et al, 2006):

- A top-down approach is defined in the cases in which a company deliberately designs and introduces a platform strategy and bases their products on it. It can also be described as based on market needs rather than existing products, since it is not necessary for the variants to be known at the time of developing the platform (Alizon et al., 2007).

- A bottom-up approach exists when the process occurs in the opposite direction, where after years of developing different products, the engineers realize that

they share, or can potentially share common systems and components and the platform emerges almost in a natural manner.

Top-down approaches are naturally suitable for cases where the variants are not well defined, in particular because the platform is intended to last for a long time during which new products will be developed.

On the other hand, bottom-up approaches are more suitable for evolutive cases, where some products already exists and it is not realistic to re-design a completely new family from scratch.

In terms of how the product range is structured to cover the market segments, there are four different possibilities (De Weck and Suh, 2003):

- No leverage: designed for a single market segment.
- Horizontal leverage: used across products of different brands placed in a similar segment.
- Vertical leverage: used across products of the same brand placed in different segments.
- Beachhead: a combination of horizontal and vertical.

According to their architecture, platforms can be (Simpson et al., 2001):

- Scale based, where the different variants are designed by scaling up or down the non-platform variables.
- Module based, where different modules are added to the common platform to form the different variants.

Process to design a product family

This subsection presents a general summary of a typical process to design a product range, it is a very high level description and each of the stages could be further

broken down and detailed (Otto et al., 2016), but this top level view should clarify the process at a level appropriate for understanding the remainder of this thesis. It is not claimed that this is the only process by which a product range comes to exist, but it is common and representative of the kind of steps required.

When a company wants to develop a product range, first they need to be clear about what are the applications for which they want to offer a product and identify the intended market.

Once the market is identified, the term *product clustering* or *market segmentation* refers to the separation of the different product market niches and the definition of the products that will target each segment (Pirmoradi et al, 2011). Although the two terms are not exactly synonymous, both refer to a similar stage in which a large set of intended applications is mapped to a reduced set of products. The generation of that map is also called *product portfolio positioning* (Pirmoradi et al., 2014). Typical existing methods to design a product platform and family assume this stage is already done and the products are well defined, whereas the method presented in this thesis will show the products as a part of the result.

The next step is to decide on the family architecture, i.e. what components can be shared across what product variants, or what will the platforms look like and how many of them will be. This is referred to as *platform identification* (Chowdhury et al, 2011), and can be a priori if it is done before designing the variants or non a priori if done at the same time.

Each variant needs to be designed, taking into account the common and the individual parts. In two stage platform design methods, all the variables of the common components are determined or optimized first, and then the same is done with the individual components of each variant. Whereas in single stage methods, all the variables are determined at the same time (Simpson and D'Souza, 2004) (Dai and Scott, 2006). In practice, this step takes many months or years, as it involves

the traditional design and development of each product variant, the methods for design considered in this thesis, including the novel method, are all high level, and do not attempt to return complete detailed designs.

During the development of each variant, there are continuous assessments of the performance of the products, and when it is not satisfactory, that leads to redesign. This is expected as part of the normal product development process. How far back the redesign goes depends on the careful thoughts put into the first stages, or how the uncertainty was managed. This whole process as outlined above is graphically shown in Figure 2.3.

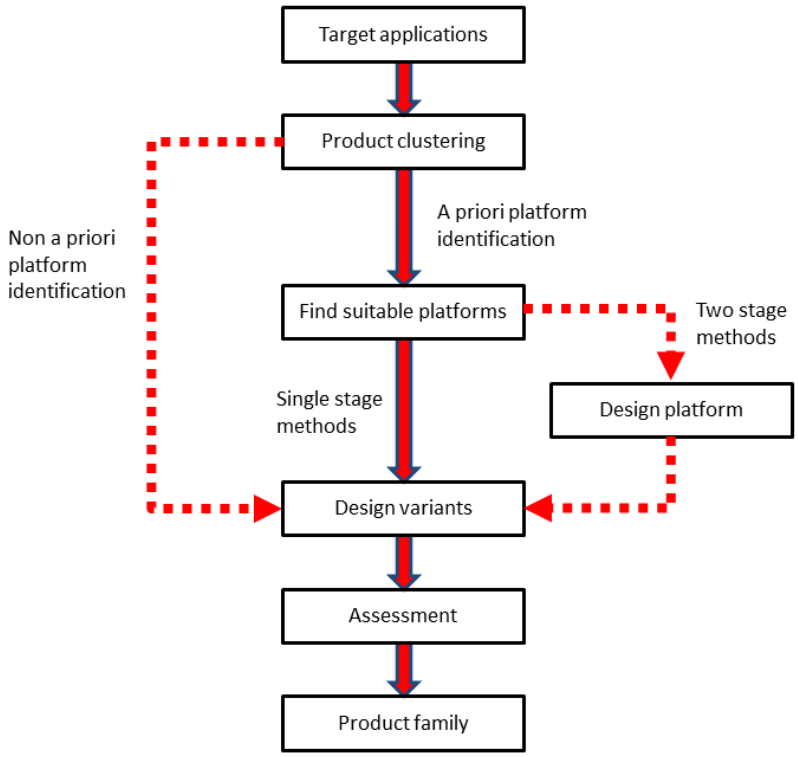


Figure 2.3: Product family development process flow

Challenges of product family design

The process described in the previous section poses a number of problems or challenges. Jiao et al (2007) classified those challenges in three main groups:

- Front end issues: translation from customer needs to product functionality.
- Family design issues: map from the functions identified in the front end to design variables.
- Back end issues: collection of issues related to processes such as manufacturing or supply chain.

Methods to design a platform architecture

The problem of product family design involves selecting the components that can be made common, fix the variables that define those components, and also the variables for all the individual components that will be part of the finalized product variants. Several methods have been developed in the last 20 - 30 years to solve that problem. They can be classified into one and two stage methods as mentioned in the previous section. One stage methods are normally reserved for families with relatively low number of products and variables, as the computational complexity increases exponentially with those numbers and the problem quickly gets out of hand. On the other hand, complexity of two stage methods only increases linearly with the number of products and variables, which make them more practical for large problems (Dai and Scott, 2006).

Most of the methods described in the existing literature belong to the second group, the following list shows a selection of them. The list is not exhaustive as this type of approach is not the focus of this thesis. The most prominent methods have been selected for the purpose of giving an overall view of the research in that field.

- *Design for variety*(Martin and Ishii, 1996). Uses two indices for the dependency between components and the amount of redesign eventually required for future variants, the purpose is to select the common components that minimize those two indices. The method is shown applied to a family of water coolers (Martin and Ishii, 2002). This method is particularly good when the long view and having a flexible platform is more important than the immediate products, and it requires a good estimation of the likely future re-work.

- *Method for architecting product platforms* (Gonzalez-Zugasti et al, 2000). Iterative process based on identification of possible platforms looking at different products variants. This method focuses on platform identification from already existing product lines, it is a clear example of a bottom-up approach to platform strategy.

- *Multi-criteria optimization in product platform design* (Nelson et al, 2001). Compares the best possible variants under a commonality constraint with those variants without the constraint. Then selects as common parts those that resulted in the closest comparisons.

- *Product platform concept exploration method* (Simpson et al, 2001). Uses surrogate models to predict product performance and then formulates a compromise decision support problem. This method requires products for which the performance under evaluation is well behaved, i.e. it is suitable for scalable variables and products with a relatively low degree of interdependence, as the surrogate models can easily break as soon as discrete components or complicated machines are involved.

- *Variation based method* (Nayak et al, 2002). Defines and minimises a function that incorporates the constraints.

- *Product family strategy and platform portfolio optimization* (De Weck and Shu, 2003). Defines product variants as vectors, whose elements are the variables, and maps them to performance vectors to compare them with the objective vectors defined by the performance of the market leaders. That map between the variants and performance vectors can be complicated and is one of the main points of this thesis.

- *Flexible product platforms* (Shu, 2005). Focus on design for flexibility to accommodate future requirements.

- *Proactive platform modularity* (Hirshburg and Siddique, 2014a). Identifies product requirements using customers surveys and selects the platform with an analysis of common functions implemented with scalable parts. Then this method uses a mathematical model to scale the platform for all the products. The method is demonstrated on a family of lawn blowers (Hirshburg and Siddique, 2014b)

An inherent problem with two stage methods is that, as some variables or components are fixed before looking at the others, this approach restricts the number and quality of possible solutions, especially in real cases of complex products with many performance attributes and where each attribute depends on non-obvious combinations of the chosen components. One stage methods have the potential to find a superior solution or set of solutions by searching the entire design space as opposed to a restricted version of it, but computational constraints and difficulties have kept them somehow in a second front of research. The objective of this thesis is to develop a method applicable to any general case, and hence the chosen approach is single stage; while keeping it computationally feasible.

Several methods were found in which both the platform and the variant variables are determined at the same time, or one stage methods. They are discussed in the

next list and explained why they still miss the capability to tackle a problem such as the case study in this thesis:

1. Effective platform family design using physical programming and the product platform concept exploration (Messac et al., 2002)

2. Effective product family design using preference aggregation (Dai and Scott, 2006)

3. Selection-integrated optimization (Khire and Messac, 2008)

4. Comprehensive product platform planning (Chowdhury et al, 2011)

5. Multi-objective genetic algorithm with generalized commonality (Khajavirad et al., 2014)

6. Platform design variable identification using multi-objective particle swarm optimization (Moon et al., 2014)

The first method (Messac et al., 2002) uses physical programming, which assigns a degree of desirability to each performance attribute. However, this method is still based on performance only, and does not represent the trade-offs involved in the design of a real product family.

The second method (Dai and Scott, 2006) uses fuzzy logic for aggregating the several performance attributes of the products, hence converting an original multi-objective problem into a single objective one. It optimizes both platform and individual variables at the same time, but assumes which components are common and which are not, i.e. it incorporates a priori platform identification, which imposes restrictions that this thesis is attempting to overcome.

The third method (Khire and Messac, 2008) applies a genetic algorithm to solve a multi-objective adaptive optimization problem. However, it reduces the multiple objectives to a single objective with an aggregation function, and hence the approach

is not truly multi-objective. The paper considers the field of platform design as analogous to adaptive systems, in which there are some fixed variables for all modes of operation (akin to platform variables) and flexible variables different for each mode of operation (akin to product individual variables). The similarity between the two fields has its limits, and problems in which different product variants carry different components instead of similar components with scalable dimensions, as the typical problem object of this thesis, are out of those limits.

The fourth method (Chowdhury et al., 2011) performs both platform identification and optimization against an objective function that aggregates performance and cost decay depending on commonality and production volume. This is the most comprehensive method found, does not have a priori decisions on commonality and allows for platforms and subplatforms to appear during the solving process. However, it is only shown with a relatively simple case and it is not clear how it transfers to a more complex case.

The fifth method (Khajavirad et al., 2014) proposes a multi-objective genetic algorithm to solve the platform problem in two levels: with the upper level looking at platform architecture and the lower level at optimizing each product variant. It uses the CI commonality index introduced by Martin and Ishii (1996) and measures the deviation from performance target values. The CI is a measure of components used, it does not take into account the relative importance of reducing one component or other, this method assumes that less components is always better, and this assumption does not always work for real industry cases.

The sixth method (Moon et al., 2014) aims to assess different degrees of commonality in the value of some design variables considering two objectives: a sum of the deviations from performance targets and the variation in the design variables. This method uses a multi-objective particle swarm optimization algorithm to find a Pareto front showing the trade-off between the two mentioned objectives. Measur-

ing a performance objective as a sum of the deviation from targets and defining a commonality objective comes with three downsides:

- It does not take into account the non-linearity of the function relating the deviation with the desirability, i.e. in some cases a deviation of 5% may be a very good result while a deviation of 10% may render a product unacceptable.

- It does not represent a realistic balance between different performance attributes, i.e. the deviation from one performance target may be far more important than the deviation from another.

- A lower degree of variability is often related to lower costs, but this is not always the case.

The last four methods consider a problem where the variables are either continuous by nature or can be relaxed and approximated as continuous. These methods are in principle not applicable to the case where the products are designed based on sets of possible components, with parts such as an engine, where any attempt to relax the problem is doomed to fail.

All this existing and ongoing research, summarised in table 2.1 provide an idea of the complexity of the problem, both in terms of definition of the objective and size. All methods are shown on relatively simple cases and there is no method that can claim to completely solve a complex real case in a practical manner. This thesis attempts to show a single stage, multi-objective method to solve a general problem that can either contain discrete or continuous components while taking into account meaningful measures for the objectives.

Table 2.1: Summary of methods for product platform design

Method	**Reference**	**Stages**	**Platform identification**
Design for variety	Martin and Ishii, 1996	2	a priori
Method for architecting product platforms	Gonzalez-Zugasti et al, 2000	2	a priori
Product platform concept exploration	Simpson et al, 2001	2	a priori
Multi-criteria optimization in product platform design	Nelson et al, 2001	2	a priori
Variation based method	Nayak et al, 2002	2	a priori
Product family strategy and platform portfolio optimization	De Weck and Shu, 2003	2	a priori
Flexible product platforms	Shu, 2005	2	a priori
Proactive platform modularity	Hirshburg and Siddique, 2014	2	a priori
Effective product family design using physical programming and the product platform concept exploration	Messac et al, 2002	1	a priori
Effective product family design using preference aggregation	Dai and Scott, 2006	1	a priori
Selection integrated optimization	Khire and Messac, 2008	1	a priori
Comprehensive product platform planning	Chowdhury et al, 2011	1	non a priori
Multi-objective genetic algorithm with generalized commonality	Khajavirad et al, 2014	1	non a priori
Platform variable identification with MOPSO	Moon et al, 2014	1	non a priori

2.1.3 Summary and conclusion on product architecture and platform design

Table 2.2 shows a summary of the different types of platforms available according to different points of view and approaches to define them.

Table 2.2: Summary of platform types and approaches

Type	Description
Two stages	Method to design a product family by first determining the platform elements or variables and then the variants
Single stage	Method to design a product family by determining platform and variant elements or variables at the same time
A priori platform identification	Method in which the common components are predetermined
Non a priori platform identification	Method in which the common components are kept open until all the elements or variables are determined
Top down	Strategy to design a common platform first and the variants later
Bottom up	Strategy to come to a platform based on existing variants
Scale platform	Family where the variants differ by a factor of scale
Module platform	Family where each variant is composed of different modules
No leverage	Platform used for products of one brand in one market segment
Horizontal leverage	Platform used across products from different brands in the same market segment
Vertical leverage	Platform used for products from one brand targeting different market segments
Beachhead	Platform used across products from different brands targeting different market segments

Different approaches exist, and it is not possible to identify a superior one independently of the problem. The decision for the approach taken for this thesis is based on the case under study and its particularities.

The problem targeted by this thesis consists of finding a method to choose the right components for a product family, while presumably some of those components will be common across all or some of the products forming platforms and subplatforms. The focus is on the parts that can potentially be made common. Part of the design of each product can never be made common, such as the chassis of a big and a little forklift truck, and hence all those parts will be out of the scope.

Two stage methods are ideal when the components that can be made common are obvious, such as the battery in a family of hand held power tools, or the shaft in a family of electric motors. However, when the ideal strategy for commonality is not clear, two stage methods unnecessarily restrict the design space, running the risk of missing potentially better solutions that are automatically excluded at the time of choosing the common components (Messac et al., 2002).

Single stage methods ensure that all solutions can be considered according to their merits, and for general problems and those such as the one that will be the subject of the case study they are preferable if they can be managed. This thesis will attempt to implement a single stage approach.

2.1.4 Existing gap

This review showed that platform design is a powerful approach to design a product line that offers solutions for several applications while containing costs and development efforts at the same time, something that would not be possible when designing one product at a time independently of each other. The way in which a product line can be structured has a large number of solutions as a result of compounding the many options for many variables, i.e. how many products, what limits to cover each product, what parts can be made common, what products will share those parts, what specifications the common parts can have and ditto for the individual parts. Finding the most suitable solution is a matter of balancing the performance on the

diverse attributes achieved by each product and the associated costs to design the entire product line. It is hard to define clear measurements that can be used to classify the different solutions and put them in order of preference. Several methods exists but not a single method that can be considered as definite.

Most methods attack the problem in two stages, with the associated shortcoming stated before. This is due to many authors considering that the downsides of missing some potentially good global solutions when introducing the necessary a priori constraints are far outweighed by the increase in tractability and decrease in computational complexity associated to two stage methods. But the purpose of this thesis is to find the best possible solution with as few restrictions as possible, and hence two stage methods are out of the scope.

Some single stage methods were found, however they either assume common parts, ignore the trade-off between performance and cost or consider continuous problems. This last point is important, since a problem in which the variables are continuous, or can be approximated as continuous, can be addressed with a number of methods such as gradient search or linear programming, however this is not the case for industrial trucks where there is a choice between one component or another and their behaviour can change dramatically.

In addition, all methods reviewed start from a predetermined set of products to design, leaving apart the problem of structuring the product line-up. This is better explained in figure 2.4. The four diagrams show the different approaches:

- Two stage methods (top-left): product variants are assumed and components are a priori classified into platform or individual parts. Platform is optimized first and then variants with the platform component already fixed. These methods are appropriate when it is clear what parts should belong to the platform and what parts should not, and it is a priority to optimize those belonging to the platform, for example, a platform intended to last for a long

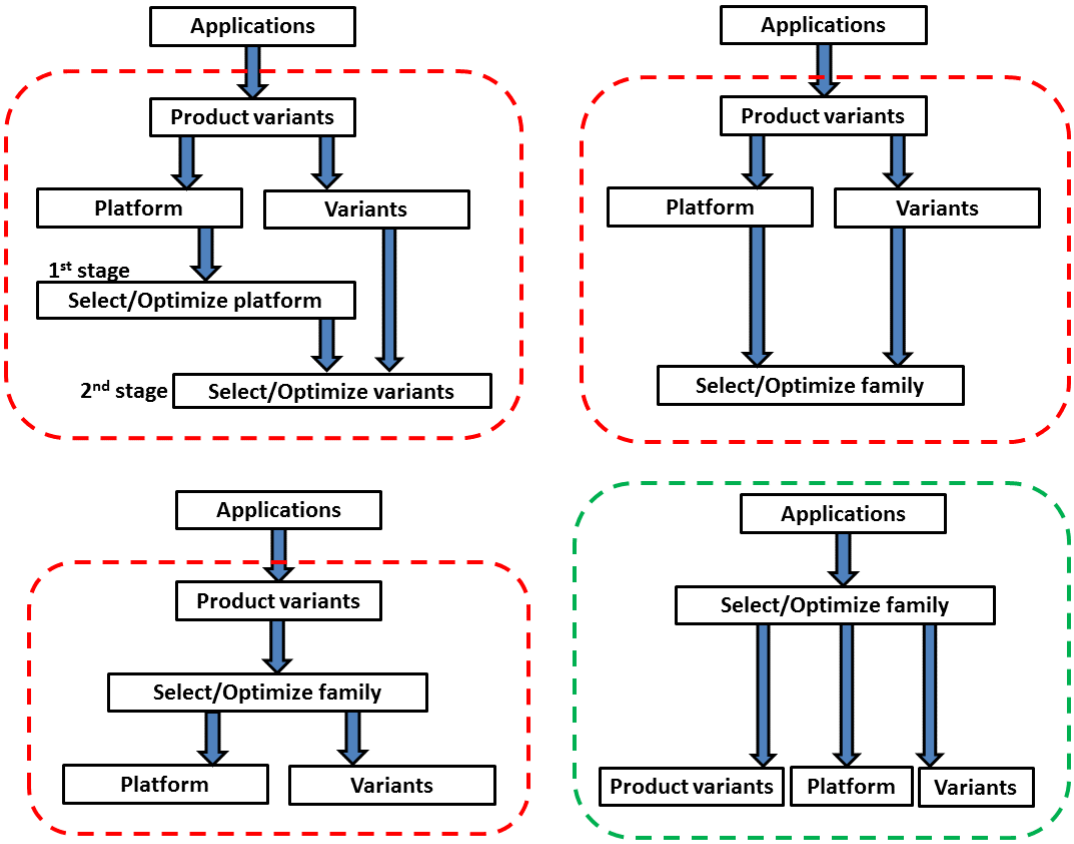


Figure 2.4: Different platform design approaches

time and multiple future products to be designed around it. Examples of this approach are all those methods identified as 2 stage methods in the section 2.1.2 *Methods to define a platform architecture.*

- Single stage a priori platform (top-right): single stage method, platform and variants are optimized at the same time but which components are part of the platform and which are individual for each variant is predetermined. This approach is also appropriate for cases where the platform is obvious, but its optimization does not take priority over the variants. Examples of this approach are the first three single stage methods identified in the section 2.1.2 *Methods to define a platform architecture.*

- Single stage non a priori platform (bottom-left): single stage method, the improvement respect to the top right approach is that platform and individual parts are not identified a priori, but a result of the optimization process. These methods can return null platforms, subplatforms, etc. That improvement also results in increased computational complexity. This is an open approach for cases where common parts are not an obvious choice. Examples of this approach are the last two methods identified in the section 2.1.2 *Methods to define a platform architecture*.

- Single stage application based (bottom-right): proposed method in this thesis, products variants are not predetermined. What is predetermined is the set of applications for which a product needs to be designed, the number and type of each product is a result of the optimization process. Product clustering, platform and variant design is integrated in one stage, unlike any of the existing methods found in the literature. This is appropriate for general cases and a superset of the other three approaches.

Apart from the process to select the components and the products, another important point is the objective for them. Figure 2.5 shows the different approaches to define the design objective for a product family independently of the method chosen to solve it.

- Dark grey arrows: The design is based only on performance. This approach is valid for methods in which the platform components are defined a priori, since otherwise an exclusively performance based objective would normally result in null platforms. It is assumed that the choice of platform is good to fulfil the cost targets, hence this approach is typically used on products for which the platform structure is obvious either due to the nature of the particular case or experience.

- White arrows: Performance and costs are balanced in an a priori articulation of preferences function, converting the original multi-objective problem into a single objective one. Examples of this approach is value driven design as defined in section 2.2.1 - Decision making theory/Value driven design, or any other combination of preferences codified in a function.

- Light grey arrows: Multi-objective approach, the problem is optimized for performance and cost at the same time returning a Pareto set (or approximation of) containing possible solutions. Preferences are then articulated over that set. This approach has the advantage of presenting several alternatives without being constrained by a preference function that may not accurately reflect the desires of the decision maker.

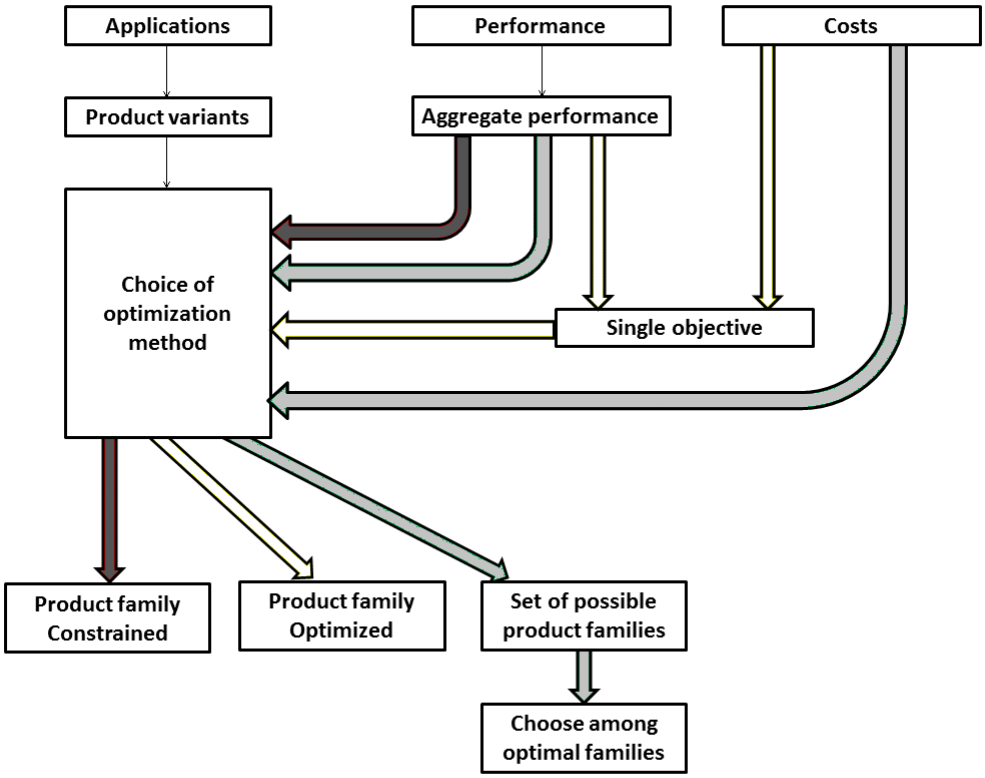


Figure 2.5: Different objective setting

2.2 Part 2: Background

This part introduces some basic and necessary concepts to understand the method that will be presented in subsequent chapters. They are:

- Decision making theory: necessary to justify the multi-objective approach in this thesis.
- Fuzzy mathematics: this technique will be used to assess different objectives measured in different units.
- Modelling and simulation: a fundamental concept for this thesis as the algorithm is based on performance values calculated through mathematical models.

2.2.1 Decision making theory

Decision making is a process by which a person, group or other entity named the decision maker selects a choice among a number of candidates. This type of processes can be found everywhere, economics, managerial sciences or autonomous driving just to cite a few examples.

Applied to engineering, Scott and Antonsson (1999) defined the problem as *Given several performance criteria which are to be simultaneously optimized, determine a method for comparing any two design alternatives that depends only on the values of the individual criteria for each alternative*

In the same paper, it is stated the difference between problems in which *best* refers to a single measurable objective and those in which the objectives are more than one. Those problems with more than one objective are referred to as multi-criteria decision making, or multi-objective optimization. Although those two terms are often used interchangeable, they are not synonymous. The former refers to the process by which a decision maker chooses a preferred option among several alternatives and does not necessarily imply optimization, whereas the latter refers to finding a number of solutions that maximize or minimize the defined objectives (Thiele et al., 2009). There are three approaches to multi-criteria decision making depending on when the decision is incorporated (Purshouse et al., 2014):

- a priori: the decision maker preferences are incorporated before the search, and hence they are an integral part of the objective function. Under this approach, a multi-criteria problem is turned into a single criterion problem at all effects.
- a posteriori: the preferences are articulated over a set of possible solutions found without them being part of the search, generally a Pareto set of solutions. This term will be defined in a subsequent section.
- interactive: the preferences are progressively incorporated during the search.

There are two main ways in which the decision maker can make their choice: satisficing and optimizing. The former refers to finding a choice that is good enough in any aspect in which the design is to be measured, whereas the latter refers to finding, or approaching the *best* option. These two approaches often appear together in problems with more than one objective, defining a problem such as optimizing objective x such that criterion y is satisfied. Criterion y is a constraint that needs to be satisfied but not optimized.

This thesis is mainly concerned with optimizing, and hence with what *best* means and how the choice leading to it can be found.

Multi-objective optimization

A multi-objective optimization problem is, as its name suggests, a type of optimization problem in which there are more than one objectives or attributes to optimize. A formal definition is (Marler and Arora, 2004):

$$\begin{aligned} \underset{x}{Minimize} \quad & \mathbf{F}(x) = [F_1(x), F_2(x), ..., F_k(x)]^T \\ & subject\ to\ g_j(x) \leq 0,\ j = 1, 2, ..., e \end{aligned} \tag{2.1}$$

It is customary to describe an optimization problem as a minimization problem. In practice many functions have to be maximized to find the optimum, profits is an obvious example, but those functions can always be converted into minimization problems for consistency by taking the inverse $1/F_i$ or adding a minus sign. When x depends on more than one variable, the problem is referred to as *multi-objective multi-variable* optimization. In that case, x is typed in bold and represents a vector composed of several variables.

A theoretical optimal point is called *utopia point*, and is that one for which all

the F functions are optimal

$$\mathbf{F}^o \textit{ is an utopia point } iff \ F_i^o = \underset{x}{minimum}\{F_i(\mathbf{x})|\mathbf{x} \in \mathbf{X}\} \tag{2.2}$$

The utopia point can rarely be achieved. When it is, that will be the optimal solution for the problem. However, in most practical cases that point is only theoretical, and hence multi-objective optimization problems generally do not have a unique solution. An important concept is that of *Pareto optimal*. A Pareto optimal is a solution so that there is no other that improves one of the objectives without making any other worse (Kasprzak and Lewis, 2000). The set of Pareto optimal solutions is known as the Pareto front.

$$\begin{aligned} \mathbf{x}^* \in \mathbf{X} \textit{ is Pareto optimal } iff \nexists \ \mathbf{x} \in \mathbf{X} \textit{ such that } F_i(\mathbf{x}) \leq F_i(\mathbf{x}^*), \forall i \\ \textit{and } F_i(\mathbf{x}) < F_i(\mathbf{x}^*) \textit{ for at least one function} \end{aligned} \tag{2.3}$$

A weakly Pareto optimal is a solution so that there is no other that improves all of the objectives at the same time.

$$\mathbf{x}^* \in \mathbf{X} \textit{ is weakly Pareto optimal } iff \nexists \ \mathbf{x} \in \mathbf{X} \textit{ such that } F_i(\mathbf{x}) < F_i(\mathbf{x}^*), \forall i \tag{2.4}$$

A point that is better in at least one objective than other point and not worse at any other objective is said to *dominate*. It is obvious that for any non-Pareto optimal solution there is always at least one Pareto optimal solution that improves it, hence the best solution for any optimization problem, independently of the criterion, is always a point contained in the set of Pareto optimal solutions. The problem is that the set of Pareto optimal solutions can be large, or infinite for continuous problems, and no particular point of it is superior to any other until a preference is defined. Several methods exist in the literature to select the best solution according to the decision maker preferences. All methods can be classified as either Archimedian

or preemptive (Messac, 1996). Archimedean refers to those methods that use a weighting of some kind for the different objectives, whereas preemptive refers to methods that optimize each objective sequentially in order of objective priority. Another way to classify the methods is according to whether the preferences are articulated a priori, a posteriori, or not articulated / interactive (Marler and Arora, 2004) (Wang et al., 2017). Some of the salient methods are briefly described in the next subsections.

Some examples of searching algorithms will be described, but an important point to consider is any of the several versions of the no free lunch theorem (Wolpert and Macready, 1997), which states that no algorithm performs better than others across several classes of problems, i.e. the best choice for a particular problem will not be a good choice for all other problems.

A priori articulation of preferences For all these methods, the decision maker specifies preferences such as goals or weights for each objective. Most methods in this category consider a utility function that can be optimized and provide a clearly ordered criterion to select among the possible solutions. This function reflects the preferences between one solution or other, and can be cardinal or ordinal (Read, 2004). The first case is a quantifiable function that provides information not only about the preferability of one solution over other, but also a quantification of that preferability. While on the other hand, an ordinal utility function only compares solutions by pairs without quantifying the difference.

It is important to highlight that when a-priori methods are used, the problem is no longer multi-objective from a mathematical point of view. All the complexities associated with true multi-objective optimization problems in terms of choosing the solutions and showing them to the decision maker or user disappear, and the resulting problem is virtually indistinguishable from a standard single objective optimization one. There is no need to or even sense in finding a Pareto front in these

cases.

Methods to define the utility function include weighted exponential means such as

$$U = \left\{ \sum_{i=1}^{k} \omega_i^s [F_i(x) - F_i^o(x)]^s \right\}^{1/s} \tag{2.5}$$

Where s is a parameter to specify the relative importance of the outliers. Values of $s < 1$ result in high valued functions not having a big effect and values of $s > 1$ result in high valued functions affecting heavily the utility function, or in other words, allowing some good functions to compensate for other poor functions (Dai and Scott, 2006). For $s = 1$ the utility function is the well-known simple weighted mean. One problem with this type of utility functions is that the different F_i can be measured in different units, and the orders of magnitude can also vary. This means that the sum may not make real sense. This can be corrected by transforming the functions F_i into other dimensionless functions that can be compared. One of those transformations is referred to as *normalization* and returns values between 0 and 1 (Koski, 1981).

$$F_i^{trans} = \frac{F_i(\mathbf{x}) - F_i^o}{F_i^{max} - F_i^o} \tag{2.6}$$

The weighted Tchebycheff method consists in looking at the higher term of the weighted mean of different solutions and selecting that one for which that term is minimum. In the lexicographic method, the functions F_i are ordered in a rank of importance and then optimized sequentially (Marler and Arora, 2004). This method imposes progressively more severe constraints as the functions get lower in the rank.

Goal programming consists in selecting targets for each function, and then minimizing the sum of the deviations. As for the weighted mean methods, the deviations can be transformed functions so the sum makes sense (Charnes and Cooper, 1977).

A posteriori articulation of preferences These methods consist in finding or approximating the set of Pareto optimal points and then have the decision maker

choose the preferred solution amongst them. One of the main advantages is that the decision maker can change the criterion without the need to recalculate the entire problem again, as the Pareto front is already available. This makes these methods more robust when a clear and undisputed preference function is hard to define.

Several methods exist for the generation of the Pareto front, some consider solving the problem for a parametric preference function and then vary the parameters to obtain different points of the front. Examples of this approach are physical programming (Messac and Mattson, 2002), normal boundary intersection (Das and Dennis, 1998), or normalized normal constraint (Messac et al., 2003). Other approach to find the Pareto front is using multi-objective evolutionary algorithms such as genetic algorithms (Deb and Sundar, 2006).

The result of all these methods is a set of solutions among which the final solution should be chosen. By definition there is no advantage in choosing any solution that is not part of the Pareto front. However, there is still the question of which of those solutions is best, and at that point a preference has to be articulated, but it is generally easier and more intuitive for the decision maker to do that when visually presented with real possible solutions rather than before knowing what the solutions look like (Wang et al., 2017).

Another advantage of a posteriori methods is that it is compatible with the concept of set-based design concurrent engineering, a design framework in which a set of possible designs are considered in principle and then elements of the set are progressively discarded as the design advances (Sobek et al., 1999). This methodology has been applied for platform based families (Levandowski et al., 2014)(Raudberget et al., 2015) and the initial set can be the Pareto result of a multi-objective search.

Interactive articulation of preferences This method consists in refining the area where the final solution is searched after successive iterations of optimization runs and the feedback from the solutions that appear. It is a combination of the

other two in which each iteration is a restricted search of the Pareto front according to a criterion that can be modified and influenced by the results. Commonly the iterative articulation of preferences is done over the Pareto set of solutions looking at the objectives, although some methods perform that articulation looking at the design variables rather than the objectives (Hettenhausen et al., 2013). The method proposed in this thesis can either be used as a posteriori articulation of preferences or as part of an interactive search.

Value driven design

Value-Driven Design or VDD is a *framework against which methods, processes, and tools can be assessed* (Collopy and Hollingsworth, 2009, Hazelrigg, 1998). It differs from traditional design methods in that instead of working to meet a set of requirements such as dimensions, performance, etc., the task of the design team is to create a design that yields the highest score on a function that converts all the considered attributes to a scalar (Collopy and Hollingsworth, 2009).

It is typical in value driven design to use *surrogate modelling*, which is the practice of replacing realistic models that replicate the physics of the real world products with mathematical models that replicate the results of the original models. This is done when the original models are too complex and a full analysis is not feasible in terms of computational time. The idea of surrogate modelling is akin to curve-fitting, although not restricted to one variable (Collopy and Hollingsworth, 2009).

The use of surrogate models has two advantages:

- The model is more or less continuously differentiable.
- They are computationally less expensive.

The defining characteristic of Value-Driven Design is that engineers, when making design choices, select the best design rather than selecting any design that meets the

requirements, or the design that is most likely to meet the requirements (Collopy and Hollingsworth, 2009)(Hazelrigg, 1998). Under Value-Driven Design, the attributes are assessed with an objective function or value model, which gives a scalar score to any set of attributes. The biggest challenge in implementing a process within the VDD framework is generating the objective function (Collopy and Hollingsworth, 2009), which needs to take into account a large number of factors that provide the product value.

Collopy (2009b) analyses several methods candidate to be used as value model and assesses them according to three attributes:

- Truth or accuracy
- Beauty or lack of complexity
- Justice or capability to lead to right decisions

The reason for beauty, a very subjective term, being in the list is mainly for transparency. A transparent method is more reliable as the results can be better understood. Most methods use economics and currency as the unit for the value function. The surplus value models consider how far the design is from the ideal revenue if there were no competitors, i.e. considering the maximum price above which the customer would not buy that good. The VDD approach is focused on single products, and not on product families or ranges.

Value-driven design has been demonstrated as an alternative to other multi-objective optimization techniques to define the Pareto frontier for products as complex as satellites (Miller et al., 2014)

However, defining an appropriate value model that fully captures the designer preferences is difficult, and guidance on how to develop them is scarce. Lee et al. (2014) provide guidance to design appropriate value functions based on the present net value and the axioms of utility theory (von Neumann and Morgenstern, 1944).

Those axioms are:

- *Completeness:* The decision maker can express a preference between any two outcomes.
- *Transitivity:* The preferences are consistent in repeated pair-wise comparisons.
- *Continuity:* The preferences can be described by a scalar quantity.
- *Convexity:* A greater chance of a preferred outcome is preferred.
- *Independence:* The preference between outcomes is independent of other outcomes.

The paper goes on to define further characteristics of effective value models:

- *Single decision maker:* The introduction of more decision makers will result on the impossibility to define a value model. This is a result of Arrow's impossibility theorem that states that the preferences of several agents cannot be combined into a single criterion that satisfy all of them (Arrow, 1950).
- *Ranking:* The value model must return a scalar number that can be ordered.
- *Uncertainty:* The model incorporates the believes and uncertainty of the decision maker.
- *Risk preference:* The preferences for the outcome take into account that uncertainty.
- *Rationality:* The action is based on maximization of the utility.
- *Driver of value:*The driver is the net present value of profit for a company or the societal benefit for a non-for-profit entity.
- *Impact of actions:* Predicts impact of design decisions on the system and environment.

- *System behaviour:* Predicts how the system and environment interact to impact outcomes of concern.

- *Stakeholder actions:* Predict the impact of stakeholders actions on the system, environment and decision maker.

- *Stakeholder concerns:* Predicts aspects of system and environment that drive the stakeholders' decisions.

Although the value driven design approach is in use in some industries, the main difficulty to implement it lies with the complexity involved in designing the value function, and the inherent effect that once that function is designed the designers are blind to the solutions that would have fitted other alternative definitions of that function. This can be overcome by defining successive value functions and running the problem with each of those definitions, but then, which one is the *real* value function? The other downside of this approach is that this method may be unable to search non-convex regions of the Pareto front (Ehrgott and Wiecek, 2005), i.e. regions where the objectives of the Pareto points are poorer than the linear interpolation of some other Pareto points. This case is very real when combining components to obtain a design, such as the case that will be presented in this thesis.

Conclusion on decision making

Decision making is a complex social process, and any non-trivial reduction of the problem leads to missing points that can be important.

Cases such as the one presented in this thesis are inherently multi-criteria decision problems, it is not possible to select a single criteria among all the existing ones and ignore the others. The question is which of the two principal approaches to follow, either articulating the preferences a priori and define a single objective to optimize, such as VDD or pre-emptive methods; or a multi-objective approach to present the

decision maker with a set of non dominated possible solutions.

The problem with a priori methods is that it is already very hard to define a good preference function that truly reflects the desirability of products with a certain level of complexity, and this difficulty greatly increases when the design object is a family rather than a single product. Another downside, is that by articulating preferences a priori, potentially good solutions are kept invisible from the decision maker (Marler and Arora, 2004), since the search will only return a good solution in the direction specified by the preference function, ignoring other solutions. A simplified preliminary study was done following an a priori approach (Zapico et al., 2015), and the shortcomings have lead to the decision to avoid this scenario as far as reasonably possible, preferring a multi-objective approach instead. This approach allows the decision maker to compare between different possibilities, making it easier to articulate the preferences after observing what is feasible. However, the share of solutions included in the Pareto front increases dramatically with the number of objectives (Deb, 2001)(Khare et al., 2003), and for more than ten it approaches 100% of the solutions, as can be seen in figure 2.6 (Ishibuchi et al., 2008), where the x-axis is the number of objectives for a problem and the y-axis the typical percentage of valid solutions that belong in the Pareto front. This figure is based on a 500 variables knapsack type problem, i.e. maximizing the value of the items to put in a bag with a weight limit, and is representative of the expected trend, but the values for each particular problem may vary depending on the individual characteristics such as number and type of variables or relation between the objectives. For problems such as the case study that will be presented in this thesis, multi-objective optimization considering all of the objectives is unmanageable unless some objectives are combined, which represents a compromise between pure single and pure multi-objective approaches. The purpose is to present the decision maker with a legible and understandable set of solutions that can be easily visualized while at the

same time avoiding undue restrictions that would hide potentially good solutions, so the decision maker can still see what their trade-off means over real solutions.

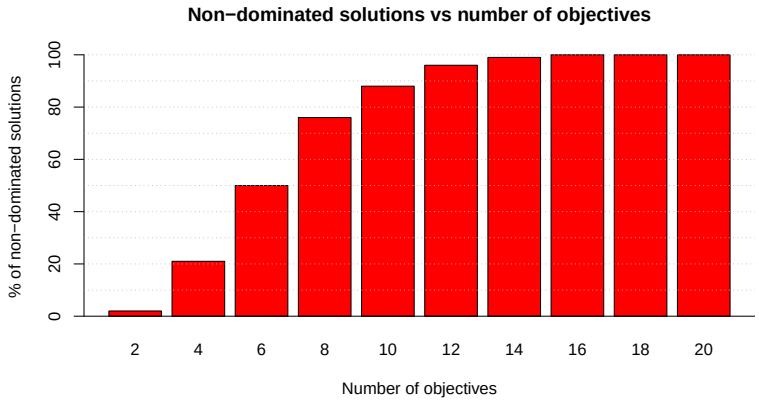


Figure 2.6: Non-dominated solutions as a function of the number of objectives (Ishibuchi et al., 2008)

Due to the restrictions imposed by the no free lunch theorem, this thesis can not recommend a particular searching algorithm for all cases. The case study will be done with a genetic algorithm but that will be a specific choice for this case.

2.2.2 Fuzzy mathematics

A short introduction to fuzzy mathematics is now presented as this is a technique that will be extensively used in this thesis to assign a score to the degree to which a product meets its performance requirements.

The fuzzy branch of mathematics includes fuzzy sets, fuzzy logic, fuzzy arithmetic, fuzzy functions, fuzzy algorithms and many other sub-fields. It was proposed by Lofti Zadeh in the 60's and has become common use in the study of systems that do not have well defined properties. A fuzzy set is defined on a domain, normally the set of real numbers, by a membership function that can take values between 0 and 1. 0 means the element does not belong in the set, 1 means the element completely belongs in the set, and any value in between means that the element somehow belongs in the set (Zadeh, 1965). Fuzzy sets are particularly suitable for representing

human assertions such as *It is very unlikely that there will be a significant increase in the price of oil in the near future.* (Zadeh, 2001) When trying to quantify that last expression there are three obvious obstacles:

- What does *very unlikely* mean? Is it 1% chance or 10% chance?
- How much is a significant increase? Is it $10 or $100
- Is there a borderline between the near future and the distant future? When?

The fuzzy approach proposes to represent the terms in fuzzy sets, as in figure 2.7

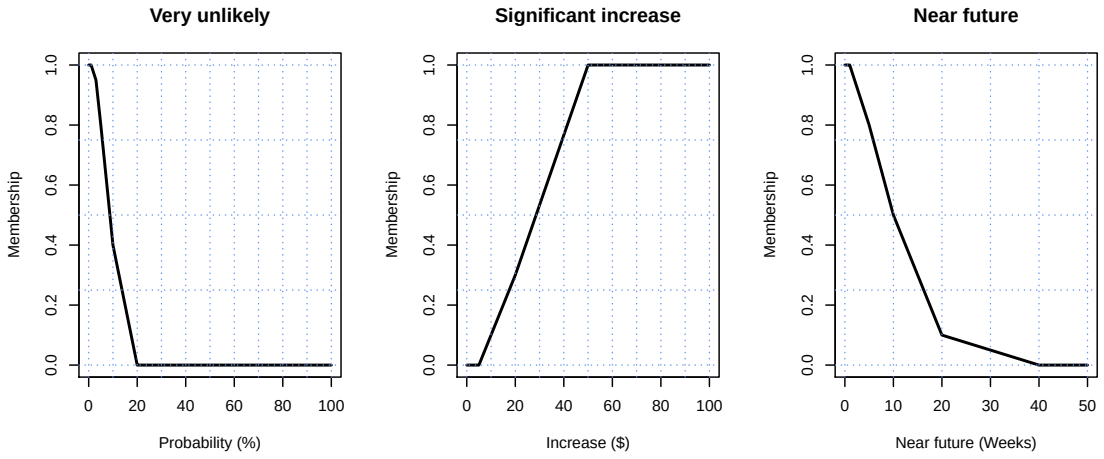


Figure 2.7: General example of fuzzy sets

Fuzzification

Fuzzification is the process of constructing and designing the functions that represent the membership to a fuzzy set (Wierman, 2010). The manual methods used for this purpose belong to two main categories (Watanabe, 1993):

- Frequency. Based on the percentage of experts that consider an element to belong to the set.
- Direct estimation. Based on ratings for compatibility between the element and the set.

In addition to these manual methods, there are also automated methods using artificial neural networks, genetic algorithms, deformable prototypes, gradient search or inductive reasoning to read large data sets and construct the membership functions (Wierman, 2010). Automated methods are more suitable for cases in which there is a large amount of data available. This is not the case in the method presented in this thesis where the fuzzy sets will be based on the ratings of a restricted number of people, and the fuzzification method to be used will be manual.

Applications related to product family design

In a review of fuzzy logic applications in product family development (Agard and Barajas, 2012), fuzzy logic has been used to define the QFD (Quality Function Deployment) to translate customer desires into design specifications. Li et al. (2016) used fuzzy arithmetic to implement expert's opinions in the definition of common, modular and scaled components, but no work has been found to use it for evaluation of the product families. Those efforts employ crisp - non-fuzzy - parameters, such as the metrics introduced by Thevenot and Simpson (2007). It has also been used as a method for aggregating information of different types in multi-objective decision making problems (Ekel et al., 2016). Li et al. (2016)

Conclusion on fuzzy mathematics

Fuzzification is the method used in this thesis to assess how a product meets performance requirements for two reasons: Firstly, it is a method that allows for aggregating and comparing different performance attributes as all are measured in the same units, those of goodness. And secondly, unlike other methods such as deviation from an ideal figure, it can incorporate non-linear relations between the physical units measuring a performance attribute and how good that is, which makes it in line with a more realistic perception of how good the attributes really are.

2.2.3 Introduction to modelling and simulation

Product modelling is a fundamental practice in engineering, it can be argued that modelling in one way or another is the main contact point between engineers and the products (Eckert et al., 2015). As this section is intended as introductory to modelling and simulation, the main questions to be addressed are what is a model? and what is a simulation?

What is a model?

There are several definitions for a model. A model is a representation of information related to a product (Maier, 2016). Or according to the US Department of Defense, a model is '*a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process.*'(DoD, 1998), whereas Frigg and Hartmann (2012) consider physical objects, fictional objects, set-theoretic structures, descriptions or equations, and combinations of them. In other words, a model is a simplification of the real event that involves a degree of accuracy that is enough for the purpose of our interest, without the need to replicate extra aspects. Models are widely used in research and development and built specially for particular purposes. An example is a clay model of a car, its intention is to provide a good visual idea of the shape of the car in 3D and sometimes be used for aerodynamic studies in a wind tunnel. A clay model is excellent for those purposes, much better than any drawings or 3D rendering on a screen for visualization, and better than a CFD model for aerodynamics. However, a clay model of a car is useless for estimating the acceleration of the car or the ride quality. For those purposes, completely different types of models are required.

There are many types of models used in engineering, such as scale models, ergonomic bucks, working mules, CAD models or mathematical models, which include dynamic models, FEA models or CFD models. They serve different purposes, some

are functional models that predict the performance of a particular system, whereas others are only intended to show the shape.

The definition of model provided at the beginning of this section distinguishes between physical, mathematical and otherwise logical models, and provides further definitions for the different types (list not exhaustive):

- *physical model*:a model whose physical characteristics resemble the physical characteristics of the system being modelled. This type includes all the scale models and different types of bucks or mules.

- *symbolic model*:a model whose properties are expressed in symbols.

- *graphical model*:a symbolic model whose properties are expressed in diagrams. This includes diagrams, decision trees, etc.

- *mathematical model*:a symbolic model whose properties are expressed in mathematical symbols and relationships

The models required for this thesis are those that are capable of relating component choice with performance. Although it is possible to use physical models, specifically scale models, to predict some performance attributes for several machines, being aerodynamic or hydraulic performance obvious cases; it is not feasible for a problem such as the one presented in this case study due to the difficulty of building scale models of the engines, or all the necessary component instances. Any other type of model not relying on mathematical relations somehow, is not capable of providing the required estimation of physical performance. And within the mathematical field, non realistic models such as surrogate models have already been ruled out due to their incapability of identifying discontinuities and singular points. For the type of problems that this thesis addresses, only models based on calculating the physics of the machines can provide the necessary information.

Most problems in Physics or Engineering are complex. The sought after result or behaviour depends on many variables and details and cannot be accurately represented with a single and elegant equation. If we wanted a perfectly accurate model, we would need to consider every single factor that has an effect on the outcome. Even for an overly simple example such as a free falling object, that would ultimately mean taking into consideration the impact of every molecule of Nitrogen and Oxygen in the air, and the position, angle and deflection of every cubic millimetre of the object, making it an infeasible problem to solve. It is also hard to decide where the limits of the system are, as the local conditions of the air cannot be isolated from the whole atmosphere as a system, which itself depends on the radiation of the Sun, etc. In practice, every mathematical model has to restrict the system boundary and restrict the effects and the granularity applied. In the case of the free falling object, the model can be something as simple as a point mass subject to the gravity law, or a mass with a surface considering the air resistance as a linear/quadratic law. This will not be an exact representation of the problem but good enough in most of the cases.

When designing mathematical models it is important to account for the effects that are relevant for the intended purpose but also to simplify or eliminate those which are not, to avoid computational burden that will slow down the simulation without adding any value. This is of particular importance in cases where the models are used for optimization, and need to be run a large number of times.

What is a simulation?

Simulating is synonymous with running a model, i.e. providing inputs to the model, calculate it and obtain the outputs. In other words a simulation is *the imitation of a real process by another process* (Hartmann 1996, Humphreys 2004). A simulation can be static or dynamic, the former consists on a single step calculation, although

there may be some parts of that calculation that involve iterations or integration over lengths, surfaces or volumes, such as for example a finite element analysis simulation. The latter, dynamic simulation, involves the simulation of a system over a period of time and, with the exception of the simplest problems, this invariably requires the use of some sort of software with a numerical integration method. Several software packages are readily available for this, some are proprietary such as Matlab/Simulink, Dymola or MapleSim, while some others such as Octave, SliLab or OpenModelica are open source, the list is not exhaustive. The models in this thesis have been written and simulated with Simulink for convenience, but this particular choice has no effect on the results. The second part of running a simulation, after writing the models, is to select a solver to integrate the models over time. This sometimes has a profound effect on the result, as some problems require particular types of solvers, and using the wrong one can crash the simulation, or in the worst case, yield false results. Solvers can be classified according to several criteria:

- *Ordinary or partial:* depending on whether the model involves ordinary differential equations (ODE), typical for multi-body mechanics problems, or partial differential equations (PDE), typical for computational fluid dynamics.

- *Fixed step or variable step:* depending on whether there is a fundamental time step, such as the case of control systems where that fixed step is the controller clock cycle, or not. Variable step solvers vary the time step during the integration depending on the rate of change of the variables. This is an advantage as it allows the simulation to speed up during periods where the system activity is low, reducing the total computation time. However, it is still fairly common to use fixed step methods for this type of problems, or at least limit the maximum possible step to reduce the errors.

- *Continuous or discrete:* depending of whether there are continuous variables,

such as almost all physical problems, or only discrete variables, such as state machines or sequencing problems.

- *Explicit or implicit:* explicit methods can be used when all the outputs can be written as functions of the independent variables either in a direct way or a cascade. Implicit methods need to be used when the output variables cannot be isolated, which is typical of bidirectional mechanical problems where there is not a clear direction of flow. Some of these problems can be rearranged to be explicit by doing the approximation of applying a time step delay in all output signals that are fed back to the model.

- *Stiff or not stiff:* depending on whether there are variables in the model with very different rates of change and frequencies.

- *Order:* this is the order of the error generated by the solver, for example a solver of order 2 means that the error tends to zero at the same rate as the time step to the third power.

Conclusion on modelling and simulation

The target of this thesis is to provide a method to guide the design of a product family by choosing the parts that can be made common to all or a subset of products. As this analysis is intended to be done at an early stage, it is fundamental to explore how each product would be depending on the selection of parts to build it. This can only be done by either experimenting with physical parts or using some kind of mathematical model. The former is expensive in terms of both money and resources, as well as limited in scope. It is not possible to test a large number of combinations physically, for the example of industrial trucks, assuming a single product with 3 possible engines and 6 pumps, this method would require to build 18 prototypes, already a challenging number. And when more parts are considered such as the mast

for different lifting heights, or considering a whole product family, the number of prototypes can easily get into the thousands, which renders this method impossible.

There are several mathematical methods to estimate a product's physical performance depending on its components. They range from simple methods such as interpolating from existing similar products or building a reduced number of prototypes, to full simulations. This latter method is the one chosen for this thesis as it allows to virtually test any combination of components not only to estimate hard performance figures but also performance through any working cycle.

This chapter has reviewed the existing literature, identified advantages, problems and different approaches as well as the existing gap that this thesis will attempt to fill. The next chapter will focus on the methodology followed to arrive to the method that will be presented in the subsequent chapters.

Chapter 3

Methodology

This section will first provide a background from the author's perspective and experience, then an overview of the research methodology theory and finally a roadmap of the methodology followed during the research.

3.1 Author's experience

The author has spend over fourteen years in R&D departments of several companies, the majority of the time in two of the top five major industrial truck manufacturers worldwide. This section is mostly a recollection of the experience acquired during that time and written in first person. That experience is fundamental for the development of this thesis as the topic and requirements arose from it. The level of detail is restricted by confidentiality and kept to a general overview.

First I worked in a department dedicated to a particular variety of truck, known as reach truck and described in the next chapter. There was an existing truck in production in 2005 and I have seen the development of two additional trucks; the first of them, called truck A, a new type of product as an alternative to the main option and the second one, truck B, a direct replacement for the main product. Both trucks were designed for different capacities, from 1.2 to 2 tonnes, and both

were designed with little regard to other products of the family, other than trying to re-use existing parts. For the truck A, many new components were designed, and the truck kept very little in common with the existing main truck, the only major component in common was the drive unit, and it was a deliberate choice to keep it. Truck B was a much closer match to the existing truck, sharing many more parts and principles. However, apart from specific cases, such as the replacement of a drive unit with a part present in a big selling truck of a different type, I have not seen too much combined effort to design different trucks as part of a bigger family from the beginning.

My role in the second company was in a team not linked to any particular product, and hence the exposure to different types of trucks was much bigger. My main mission was to build mathematical models of different systems and estimate several performance figures for alternative components and configurations. Here was where I realized the power of using simulation for assessing a large number of possibilities in a relatively fast and cheap manner. As an example, one of my projects consisted on assessing the impact on fuel consumption of fitting different kinds of engines, pumps, lifting mechanisms and drive trains. A selection of the potential configurations were based on existing products, so the results could be validated.

After that, I left the industrial truck industry but continued writing vehicle dynamics models and simulations with a well-known Formula 1 and supercars company. In that job I learnt about sampling searching spaces and running simulations with higher degrees of accuracy and more complex cycles such as the response of a racing car during a lap over a particular track, where the accuracy needs to reach the tenth of a second.

My current job with the biggest steel producer in the world has moved away from dynamic simulations but on the other hand it has made me more familiar with

searching over vast spaces and using a diverse set of optimization techniques such as genetic algorithms or ant colony optimization. This recent experience has also helped me understand better the difference between truly multi-objective and single objective optimization, and be more aware of the shortcomings of the latter. This last point has had a definite influence in the thesis, shifting the focus from developing an appropriate preference function for a product family to the generation of a Pareto set over which the decision maker will choose its preferred solution knowing what is possible and what is not.

3.2 Methodology theory

Design is a complex topic involving multitude of different aspects, and design research must account for this by being open in scope and draw from many varied fields.

The main goals of design research are understanding design and improving it (Eckert et al., 2003), being the former a pre-requisite for the latter.

Blessing and Chakrabarti (2009) divided design research in four stages, this methodology is known as Design Research Methodology or DRM.

- Criteria definition: definition of the criteria or objective by which the design will be evaluated.
- Descriptive study 1: literature and observation to understand what factors influence the identified criteria.
- Prescriptive study: development of a method to improve the current situation according to the identified criteria.
- Descriptive study 2: evaluation of the applicability of the method and the success in improving the identified criteria.

Eckert et al. (2003) proposed a spiral model for design research, which is shown in figure 3.1 and differs from the DRM in that it is a continuous process that each project can enter and exit at any pair of phases depending on the scope, does not require to identify a criteria for success as a premise, and recommends validating each of the stages rather than a unique validation at the end. The arrows in that figure leading to cells called *Evaluation of ...* could be interpreted as bi-directional to reflect possible iterations depending on the results of the evaluation.

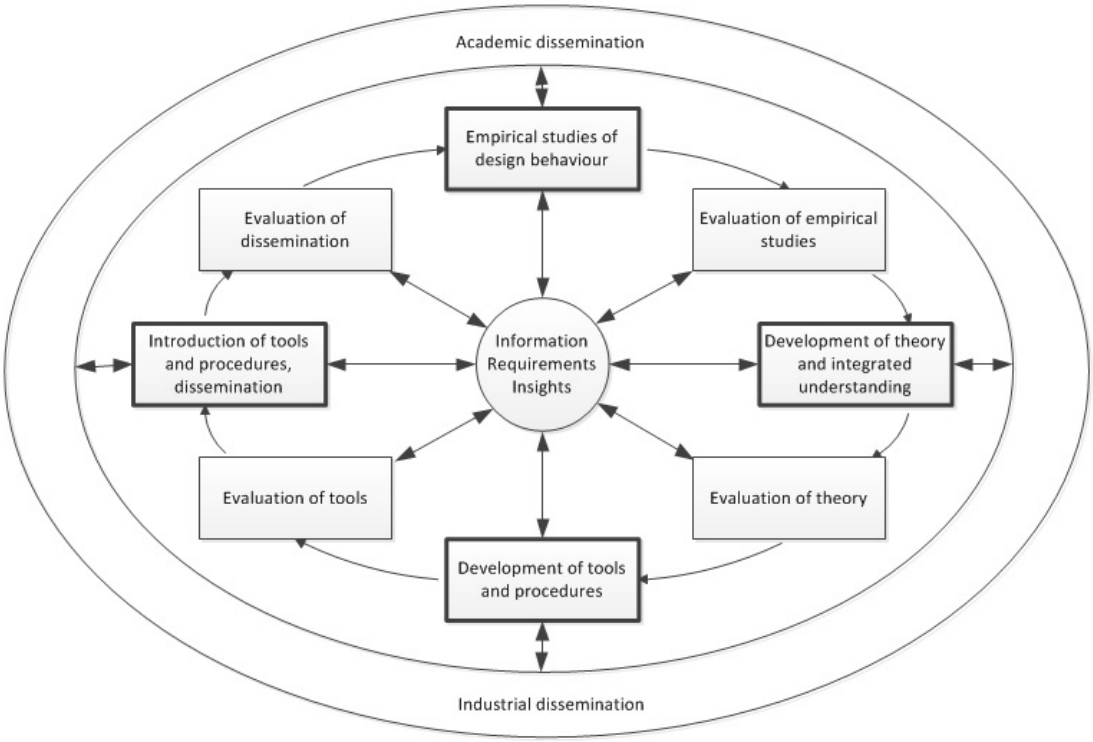


Figure 3.1: Spiral of design research (Eckert et al., 2003)

The approach followed by this project will be declared in section 3.3 research methodology.

3.2.1 Case study as a design research methodology

Case study methodologies are common in several fields, and have a long tradition in social and life sciences. They are fully discussed by Yin (2017) who defines them as

empirical studies of phenomena in their real context with which they are intrinsically associated. There are advantages and disadvantages of case studies as a research methodology (Zainal, 2007) (Yin, 2017), advantages are:

- The data is examined in the real context
- Allow for quantitative and qualitative analyses of the data
- Helps to explain the complexities of the real life scenario

And disadvantages or criticisms are:

- Potential lack of rigour due to weak methods or biases
- Difficulty to generalize the results
- Can take too long and produce too much data difficult to organise

This methodology is also used in engineering and design research, and Teegavarapu et al. (2008) argues against the main criticisms when the methodology is applied in these fields:

- Case studies are rigorous if the stages are thoroughly followed. They benefit from triangulation, or the fact of collecting data from different sources and in different processes and observe how they converge.
- Many universal theories were based on analytic generalization of single studies rather than finding statistical significance after a large number of experiments. The key is the criticality of the experiment or case study.
- Time and excessive data can be relevant sometimes but is generally not the case for design research when the case study is adequately defined.

3.2.2 Verification and validation

Verification and validation are two terms that often appear interchangeably and with blurred definitions. Pedersen et al. (2000) made the following distinction when the terms are applied to engineering research:

- *Verification*: internal consistency, i.e. whether the results are coherent with the premises.
- *Validation*: justification of knowledge claims, i.e. whether the theory is valid and useful for the problem under study.

According to the writer's experience, in the industry the terms normally denote:

- *Verification*: necessary checks and tests against the specifications to prove that a prototype or product is built up to them, this includes safety, standards and legislation requirements.
- *Validation*: necessary checks and tests to prove that a product or prototype functions as expected and desired, i.e. not only it meets the specific requirements but it also fulfils its intended functions at an acceptable level.

Pedersen et al. (2000) further distinguishes between two validation approaches:

- *Logical empiricist validation*: a formal process that categorically proves or disproves a proposal.
- *Relativistic validation*: a gradual process of building confidence in the usefulness of a new piece of claimed knowledge.

The former is the typical approach to prove a mathematical theorem or a theory in hard experimental sciences, but it is not appropriate for a problem such as that one under study in this thesis, where there is not a definite answer that can claim to be *the* solution to the problem. For this second approach Pedersen et al. introduced the validation square, shown in figure 3.2 and composed of four phases:

- Theoretical structural validity: correctness and logic of the method and the steps that compose it.

- Empirical structural validity: adequateness of the example over which the method is tested.

- Empirical performance validity: performance of the method on a case study is correct.

- Theoretical performance validity: performance of the method is robust beyond the case study.

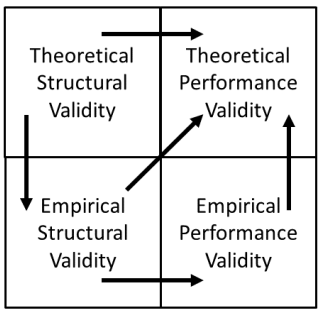


Figure 3.2: Validation square. Pedersen et al., 2000

3.3 Research methodology

This section explains the methods followed to address each of the research questions and how that fits with the formal research methodologies previously described.

There is no criteria for success, and hence the DRM cannot be strictly applied, although some of their concepts are.

Regarding the spiral of design research, this project aims to understand the benefits of platform design strategies, barriers to its adoption, propose a method to overcome them and evaluate it. Hence it can be said to enter the spiral at the phase *Empirical studies of design behaviour*, and progress through *Evaluation of tools*. Strictly speaking, this project does not follow the spiral model exactly, as the evaluation of theory and evaluation of tools are done together over a case study.

3.3.1 Research questions 1 and 2

How can a platform strategy improve product development processes? What are the barriers to industrial adoption of product platform strategies? The literature on the field will be reviewed together with an analysis of the current practice in an industrial truck company based on experience, observation and interviews. That analysis will involve a description of the industry and the design and development process. Interviews will be conducted to gain a deeper insight on the reasons for platform based designs and the barriers to their implementation.

These two questions correspond with the *Empirical studies and evaluation of empirical studies* phases of the spiral research model.

3.3.2 Research question 3

Can an alternative product platform strategy be devised that addresses these barriers? A method will be developed taking into account the problems identified in the previous research questions. The method will be described in a general manner.

This question corresponds with the *Development of theory and integrated understanding* phase of the spiral model.

3.3.3 Research question 4

How well could this new strategy perform in a real world industrial context? The method described in the previous question will be demonstrated over an industry case. This question corresponds to the *Evaluation of theory* phase of spiral model. The theory will be evaluated over a case study and that will include all the necessary work in terms of models and algorithms to run the case, so this part of the project will also touch on the phases *Development of tools* and *Evaluation of tools*. However, the tool used will be tailor made for the particular case study and it is not the intention of this thesis to provide a tool that can be directly used for any other case.

3.3.4 Software

Two different software packages will be used for this research project:

- Matlab/Simulink: a language and suite for modelling almost any mathematical problem. Simulink is a graphic environment to model and simulate dynamical systems and nearly a standard for the automotive and aerospace industry. In this project it will be used to write the models that calculate the different performance attributes of each truck candidate. There are alternatives such as Modellica or Octave, and the particular choice does not have an effect on the results. Simulink was chosen for convenience and familiarity.

- R studio: an environment for the R language, this will be used for the development and coding of the searching algorithm, as well as for the analyses included in the validation tests. This language is widely popular in the data analysis community and perfectly suited for the analysis, whereas its selection for the main algorithm is due to convenience, i.e. availability and author familiarity with it. The algorithm could be coded in any other language such as Python, Matlab, C++, etc. The choice can have an effect in the efficiency

and computation time, but a proper analysis of that is out of the scope of this project.

3.3.5 Verification and validation

Verification and validation for this project will consist of a series of tests that build confidence in the two aspects; how the algorithm is logic and consistent - verification - and how it provides useful knowledge that helps the designers - validation. The method proposed in this thesis will consist of three main tasks:

1. Definition of objectives for the product family.
2. Performance models to estimate characteristics of potential products.
3. Algorithm to search for the most appropriate solutions.

A test or sets of tests will be defined to validate each of these tasks.

Objectives validation

This step is intended to validate the objectives towards which the problem will be optimized. This is a validation task as the target is to prove that the definition of the objectives is such that the solutions provide useful information. The technique to use will be a comparison between the ranking of several solutions according to the users and the ranking provided by the objectives model.

Performance model verification

The models written to calculate the different performance attributes will be time simulation mathematical models. It is necessary to verify that those models replicate the true physics, and for that some known products will be modelled and compared to actual performance figures. This step is part of the verification process and one

that can be classified as logical empiricist according to Pedersen's classification as it is a traditional model vs experiment results comparison.

Searching algorithm verification

This is a process to verify that the searching algorithm performs appropriately and efficiently. The algorithm is heuristic and it cannot be proved how far the best solution is from optimal, but statistical tests will be used to increase the confidence in the algorithm efficiency.

Industry design validation tests

A validation exercise will be carried out over the case study and its results. It consists of a series of tests to build confidence on the different steps that compose the method. They include:

- Performance models - accuracy of the models used to estimate the different performance attributes of the different products.
- Objectives - how they reflect the preferences of the decision maker.
- Search - check that the search has produced meaningful results.
- Results - consistency of the results with the problem principles.

3.4 Summary

This chapter has detailed the methodology that will be followed during this thesis to find answers to the research questions, firstly looking at the current status in the literature and in an industrial setting; and then proposing a method to address the identified barriers and validating it over a case study.

Chapter 4

Industry justification

This chapter complements the literature review in the search to answer the first two research questions on how a platform strategy can improve the product development process and the barriers to its implementation in the industry. This chapter is mostly based on the author's experience and draws from the particular industry of materials handling trucks where the author has spend 11 years. Nevertheless, its ideas and conclusions can be generalized.

The mentioned industry of industrial trucks involves the vehicles used typically in warehouses and industrial yards to move different sorts of loads from one place to another. On a high level, those vehicles must meet two objectives to be successful: move the loads with the efficiency and speed required and be as cheap as possible to run and maintain.

This is a common and clear example of conflicting objectives, as it is almost intuitive to any casual observer that the faster and more powerful the truck is, the more expensive it will be and the more fuel it will consume.

The chapter begins with an overall description of the industrial trucks industry and its characteristics that motivates this research on platform design. The second part is an attempt at generalizing the problems found in this industry so that they can be extrapolated to a general case. Finally, the third part is a wrap up of the

chapter justifying the decisions made for the development of this thesis and leading to its core chapters.

4.1 Industrial trucks - overview

Industrial trucks are a convenient example for this research, as they are complex and diverse machines that nevertheless are all related by the fact that they perform a similar function: move a load from point A to point B. This characteristic makes them a suitable candidate for the application of common design approaches and component sharing. On the other hand, there are significant differences between different uses. The loads to be moved vary greatly in mass, size, shape and height to be picked from or stack to. The environments vary from enclosed warehouses and cold stores to timber yards, scrap yards, docks or even potentially explosive atmospheres. Hence, while the basics look similar to the untrained eye, products for different applications need to be designed with all the particularities in mind.

According to ISO 3691, the main international standard applicable to industrial trucks, a self-propelled industrial truck is *any wheeled vehicle having at least three wheels with a powered driving mechanism, except for those running on rails, designed to carry, tow, push, lift, stack or tier in racks any kind of load and controlled by an operator.* (ISO, 2012)

The fact is that there are many different vehicles that fit that description, the main types are shown in figure 4.1

- *Counterbalanced forklift trucks*: Probably the best-known for the general public. They are the kind of vehicles often seen in warehouses and industry yards with an extendible metal frame at the front and a counterweight at the back. This counterweight is necessary due to the load being carried ahead of the front axle. The metal frame at the front is called *mast*, and it serves as the guide to

Figure 4.1: Different types of industrial trucks)

the *carriage* and *fork arms*, which is the part that holds the load, which will normally be resting on a wooden pallet. This type of truck typically ranges in capacity from 2 to 40 tonnes, and can be powered by an internal combustion engine (petrol, diesel or LPG) (1 in figure 4.1) or electric motors (2 in figure 4.1), in the latter case normally one for each drive wheel. The source of energy for electric vehicles is a lead acid battery for most of the existing units, big and heavy, so apart of the primary function, they also serve a secondary purpose as counterweight. Currently there is a high focus on researching alternative sources of power for electric trucks, such as other battery chemicals or fuel cells. As suggested, the reason for that research is energy efficiency, and not mass reduction. Counterbalanced trucks can be used indoors and outdoors, with the electric versions easier to find in indoor applications. Although they are generically referred to as forklifts, not all of them incorporate forks. There are many of these trucks equipped with different attachments such as clamps,

typically found in the paper industry, a protruding shaft for steel rolled coils, or electro-magnets for a scrap yard.

- *Reach trucks and straddle trucks*: (3 in figure 4.1) Highly specialized trucks purposely built to operate in warehouses with flat floors and normally have capacities of up to 2 tonnes. The operator normally sits at an angle of 90 degrees with the direction of travel, and the trucks are much shorter than counterbalanced trucks. They normally have only three wheels, two at the load side, which for this type of truck is called rear, and one, normally the drive wheel at the front, right below the driver. This configuration allows reach trucks to turn in very confined spaces and operate in narrow aisles where counterbalanced trucks would not be able to manoeuvre. The tyres are normally made of a hard material such as polyurethane, and the heights at which they can stack a load can reach up to 14 meters, for which they have 2, 3 or even 4 stages masts. Compared to counterbalanced trucks - with the exception of container handling counterbalanced trucks -, reach trucks can typically reach higher, and hence their name, as they are designed to operate on flat well conditioned floors where it is easier to ensure their stability. A consequence of the different uses, is that reach trucks can spend up to 50% or more of their time lifting, lowering and stacking, whereas counterbalanced trucks will rarely exceed 20% in those operations, being the rest of the time driving from one place to other. Reach trucks are always electric powered, as they are intended for indoor use. One major feature that separates them from counterbalanced trucks is that the load is carried between the rear wheels, and not beyond their axis. This reduces the need for a heavy counterweight.

- *VNA trucks - very narrow aisle*: (4 in figure 4.1) This type of truck is designed for working in the homonym very narrow aisle warehouses. Their purpose is to move up and down an aisle and stack or retrieve loads from both sides.

They never leave the aisle and are not used to load a lorry or take the load somewhere else, hence they do not need to turn and are often guided by side rails along the aisle. The forks can face both sides of the aisle and the capacity is similar to that of reach trucks.

- *Tow tractors*: (5 in figure 4.1) As their name suggests, their purpose is to pull a trailer with a load. These trucks are easy to spot in airports handling the check-in baggage or in train stations pulling wheelie bins. They are also typically electric powered and can tow up to 25 tonnes.

- *Reachstackers*: (6 in figure 4.1) Also known as container handlers or variable reach trucks. They have an extending arm, called boom, rather than a mast and can either have a carriage with fork arms or a container handling device designed for 20 or 40 feet containers. The most common environment for this type of truck is the docks, and the capacity can reach 52 tonnes. They are big trucks and the driver needs to climb a ladder to reach the cabin. This is also the most expensive type of truck, with prices ranging $150,000 to $300,000

- *Rider pallet trucks*: Some look like a downsized version of reach trucks (7 in figure 4.1) whereas others have a platform behind the main body for the operator to ride on (8 in figure 4.1). They are also designed mostly to work indoors in warehouses with capacities of up to 2 tonnes, however, they are not capable of reaching equivalent heights.

- *Pedestrian trucks*: (9 in figure 4.1) The smallest type of truck, they do not have a position for the driver, he or she walks alongside. Due to safety reasons they cannot lift higher than a person height. Their capacities are similar to other warehouse equipment, up to 2 tonnes.

- *Side loaders*: (10 in figure 4.1) This is a truck commonly seen in timber yards,

with capacities around 6 tonnes, and two fork arms on one side of the truck to carry the load parallel to the longitudinal axis.

Industrial trucks are rated according to the maximum load that they can handle safely, and those loads can go from 1 tonne to 52 tonnes. In practice, this means a high variability in size.

Figure 4.2 shows the main parts of a typical forklift truck on a high level. Industrial trucks are highly configurable. even when considering only one type of a particular brand, for example a 2.5 tons internal combustion counterbalanced truck, the customer can choose among eight different engine options, three wheelbases, three track widths, three different types of mast, each one of them with heights varying from 2090 mm to 7800 mm in increments of 100 mm, eight sizes of forks, two types of carriages, three types of cabs, and many more options. The number of variations of trucks of this type that can be built without repeating the same configuration is in the tens or hundreds of millions.

In terms of prices, industrial trucks range from $150 for a hand powered unit or $4,000 for small motorized models to $300,000 for the biggest ones. A typical 2.5 ton Diesel counterbalanced truck costs around $25,000, whereas a similar truck electrically powered costs around $36,000 including the battery. Purchasing price is only a relatively small component of the total cost of ownership. Considering a typical design life of 20,000 hours, a 2.5 ton truck can imply a fuel bill between $60,000 and $120,000 depending on the tax regime of the country in which it operates. Other smaller, although significant costs include service and consumables such as tyres, chains, oil, filters, fork arms, etc. However, the main cost by far of operating a truck is the driver. For the typical design life, the operator can represent up to 80% of the total cost. This makes a high performing truck capable of moving the same number of loads in less time an attractive option, specially in areas within the developed world where labour costs are relatively high.

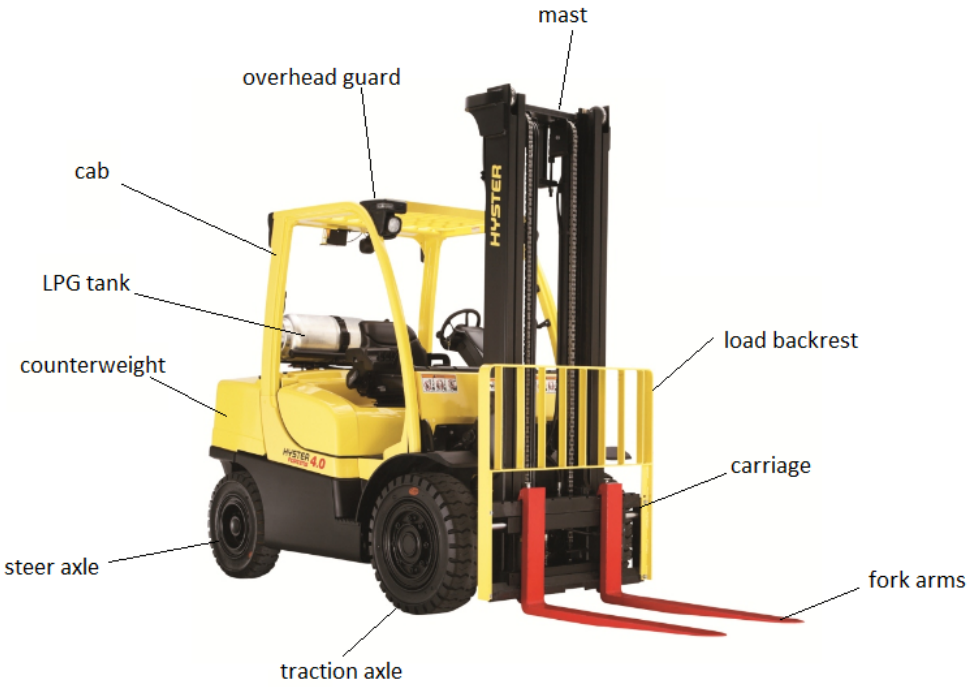


Figure 4.2: Main parts of a forklift truck

4.2 Market overview

To provide an idea of the market size, in 2012 a total of 943,724 new trucks were sold worldwide (MMH, 2013). The top five manufacturers were:

1. Toyota (Japan)
2. Kion (Linde, Still & OM) (Germany)
3. Jungheinrich (Germany)
4. NMHG (Hyster & Yale) (USA)
5. Crown (USA)

Of those, only NMHG, and Kion until recent years, offered a whole range from small pallet trucks to container handlers. The other brands are more specialized in

warehousing or low to medium capacity counterbalanced, whereas brands specialized in big trucks such as container handlers do not appear at the top due to the reduced market size. In addition to the list, there is a large number of smaller, although important and competitive, manufacturers distributed all around the world, and mostly dedicated to a specific range or type of product. The big manufacturers, such as those on the list, sell trucks worldwide and operate as classical multinational corporations, with representation, manufacturing facilities and even design teams in several countries. Due to particular market requirements, the trucks are not necessarily equal for each country, for example pallet trucks provided with a rider platform are required to be able to coast to a stop when the operator leaves the platform if they are intended for the American market, however, this feature is prohibited in trucks destined for Europe. Another example is that many counterbalanced trucks are equipped with petrol engines in America, but this type of engine in a truck is unheard of in Europe.

4.3 Product variability

Industrial trucks are highly customizable products for which the customer is presented with a large number of options. As an example, figure 4.3 shows the main available options for a 2.5 tonnes counterbalanced truck. The table shows 55776 different possible configurations for that truck, and the list is not exhaustive, as there are further options such as cab materials, heating, extra lights, CCTV, etc.

This variability is necessary since the truck needs to be adequate for the intended application, and it is not possible to build all the trucks with the same mast height for example. However, what is possible is to share technology and basic design although the final dimensions are customized. This makes industrial trucks an interesting candidate for a study on platform design and methods, with an almost unaccountable number of different applications and not clearly defined a priori limits.

Power source	Type	Wheel base (mm)	Track width (mm)	Fuel	Engine	Mast type (All)	Mast lifting height (mm)	Forks (mm) (All)	Carriage type (All)	Cab (All)
Electric	DBB	1230	905, 1035			LFL	2090, 2490:100:3790, 3930:100:5130	760 900 1000 1100 1200 1400 1500 1800	Std	No cab
		1377	905, 1035							
	CBB	1606	938, 1054							
		1750	938, 1054							
ICE	Pneumatic tyres	1623	960	Diesel	Y2.6	2FFL	2500:100:3800, 4020:100:5320			
					Y3.0					
					Y3.3				ISS	Std cab
		1700	1025	LPG	M2.0					
					M2.2	3FFL	3750:150:7800			
					M2.4					
	Cushion tyres	1430	929	LPG	K2.4					
					P2.5					

Figure 4.3: Options for a 2.5 T counterbalanced truck

4.4 Product requirements

There are two main types of requirements: legal and commercial.

4.4.1 Legal requirements

Industrial trucks are work equipment, and as such they are subject to stringent health and safety requirements. In the European Union they are covered by the Machinery Directive 2006/42/EC, which sets essential safety requirements and also assigns the manufacturer the responsibility to perform a product risk assessment and design the vehicles according to the state of the art regarding safety. In order to facilitate this task, ISO and CEN publish several standards, some of them are harmonized by the EU Commission. Harmonization means that the EU Commission accepts compliance with the standards as a means to prove presumption of conformity with the legal requirements. A little known fact, even among engineers, is that unlike the automotive industry, for industrial vehicles it is not compulsory to follow those standards. The only compulsory legislation is the Machinery Directive. However, the text in the Directive is vague and it can be hard to prove compliance with it in the case of not following the standards, specially if there is an accident involved. In North America there is a different set of standards published by ANSI,

but in recent years much work has been done to unify the requirements for all the countries, resulting in the publication of the ISO 3691 series of standards intended to cover all the world. However, even that ambitious standard still prescribes different requirements for specific markets, mainly aligned in two groups: countries following European approach or American approach. Those requirements are not design restrictive, i.e. there is no specification for any technology to implement. What the standard specifies is a set of safety performance requirements to be tested or assessed, such as minimum insulation, maximum level of electromagnetic emissions, maximum speed at which the load may fall in case of hydraulic failure, or maximum breaking distance depending on the speed. The manufacturers are free to design the vehicles and sign compliance with the standards as long as they meet the performance requirements specified in the texts (ISO, 2012).

All legal requirements must be fulfilled for a truck to be sold to a customer. They are hard constraints on the design.

4.4.2 Commercial requirements

Apart from legal requirements, a truck needs to meet other additional requirements to make it appealing to the customer. The truck needs to offer a level of performance high enough to fulfil the required application, and be competitive with the other products in the market. For example, a truck capable of lifting 10 tonnes with a maximum lifting speed of 0.4 m/s and a fuel consumption of 8 l/h cannot be sold at the same price as a competitor truck that achieves 0.5 m/s while consuming only 6 l/h of fuel. These performance requirements need to be defined by the brand and the design process should be oriented to achieve them. Normally, and industrial trucks are not an exception, achieving higher performance requires more expensive designs using higher end technologies, components and materials, so there is a balance to be found.

4.5 Product development

This section is entirely based on the author's experience with two of the major industrial truck manufacturers. The normal process to develop a new truck begins with a decision from the project review board. The decision can be triggered mainly by two reasons:

1. A perceived gap in the market not covered by the current range and with potential for profitability. That results on a new truck type.

2. A model that is in the process of becoming obsolete, either technologically, ergonomically, economically or due to impending legislation, and needs replacement. That can be an update of the old model or an entirely new truck.

The project is then assigned to the development centre specialised on the particular truck sector, which adds it to the workload according to the specified priority. The specialization of different design centres is not necessarily dependent on the geographic market for which the truck is intended. Often the reason for the specialization is legacy, as some of the bigger manufacturers acquired smaller brands dedicated to specific product types or markets. But even in the case that the development centres are agglutinated in one location, there will still be separated teams with expertise in particular products. One of the effects of this localization is that different products are typically designed independently from each other, unless there is a deliberate strategy to align them. There is a formal process consistent on several milestones between the green light for the project to start and the final day in which the vehicle is put on the market. Although each company will have their own process, the typical stages between those milestones or gates according to the author experience are showed in figure 4.4

The organisation can be akin to the lightweight project matrix organisation described in Ulrich/Eppinger (Eppinger and Ulrich, 2012), although this can also be

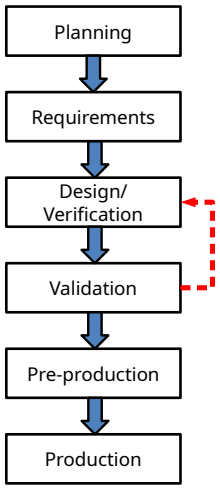


Figure 4.4: Typical design gate process

company dependent. The team assembled for a particular project is composed of representatives from several departments, including engineering, marketing, manufacturing, finance and logistics. They meet on a regular basis and always before any of the milestones is signed off. The requirements that must be met for sign off are clearly specified in an internal document. The whole process can last from one year for a small project such as adding a simple feature to an existing truck, up to three years or more for a new truck. An example of a mid-range project is illustrated in figure 4.5. The example shows a high level view, most of the mentioned tasks are further separated in several subtasks.

The main steps included in the chart for gate 1 - planning are:

- Planning: all the project planning including times, resource allocation, etc.
- Ergonomic buck: this is not a working vehicle but a wooden or tubular structure on which several elements such as seats, levers, steering wheel, etc. can be assembled to test ergonomics and driver perception.

For gate 2 - requirements:

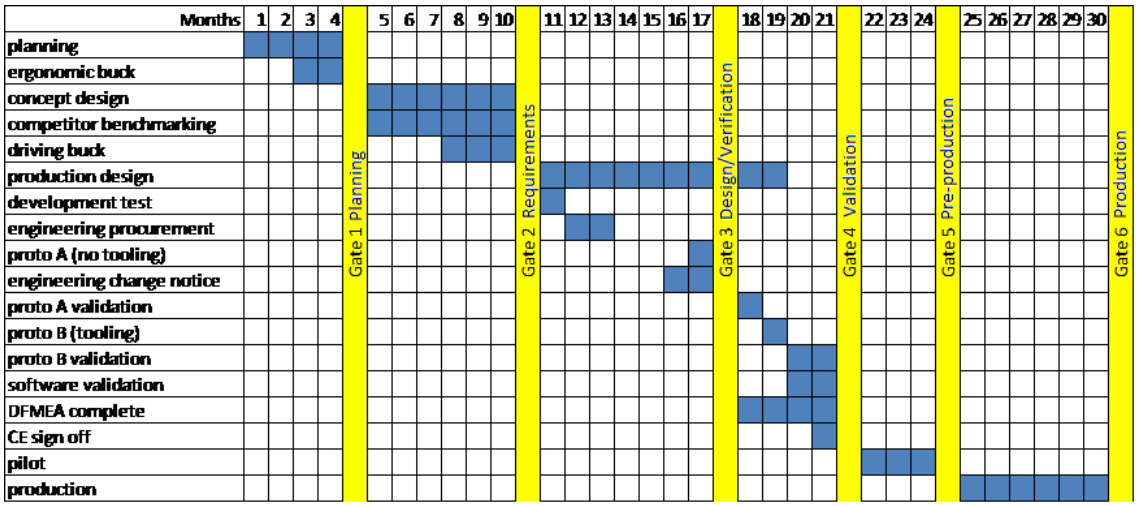


Figure 4.5: Typical design time line

- Concept design: this phase is where all the ideas for the new vehicle are generated and evaluated. At the end of this phase there is a rough idea of what the truck will look like, although not every detail is fixed.

- Competitor benchmarking: main competitor products are intensively tested in order to define the design requirements. This benchmarking is to fine tune the requirements, some level of benchmarking is also done continuously every time a main competitor releases a new truck.

- Driving buck: this is a working prototype which differs greatly from the new design, but on which new parts or solutions can be tested. Typically, this can be a modified late model truck.

For gate 3 - design/verification:

- Production design: this is a long stage involving all the CAD design, performance mathematical modelling and component specification. Depending on the project depth it can last anything from six months to over three years.

- Development test: the development team starts conducting performance and safety tests on the driving buck, as well as bench tests.

- Engineering procurement: logistics start searching for parts according to the specifications given by engineering.
- Proto A: this is a working prototype that will be very close to the final design, only minor changes are expected between the two. This prototype is build in a lab by hand, hence it is typically a very expensive vehicle.
- Engineering change notice: design is frozen, and further changes need to be documented as engineering changes from this point.

For gate 4 - validation:

- Proto A validation: the development team uses the proto A vehicle to perform all the necessary tests to ensure legal compliance and requirements fulfilment.
- Proto B: this phase involves the generation of the tooling that will eventually be used to manufacture the truck. Unlike proto A, this prototype is built in the factory and can be considered the zero series.
- Proto B validation: validation continues on proto B, as this is the final truck on which legal compliance will be signed off.
- Software validation: Software is also validated at this stage. This is one of the few things that can still have changes before production.
- DFMEA complete: together with the legally required risk assessment, this is part of the paperwork necessary to sign off the truck. This is an acronym for Design Failure Mode Effects Analysis, which is a bottom-up approach consisting on an assessment of all possible failures that may arise in any part of the vehicle, the consequences and the measures in place to prevent or detect them.
- CE sign off: declaration of conformity with the Machinery Directive 2006/42/EC.

For gate 5 - pre-production:

- Pilot: a reduced number of trucks go through the production line to check everything is in place and correct. These will be real trucks to be tested by specific customers as beta trials.

And for gate 6 - production:

- Production: the truck is finally rolled out to the factory to be mass produced.

This is an example of the design process for industrial trucks, other companies dedicated to other type of products may have different processes involving different time scales, and different iterations, although the philosophy will be similar.

The most salient point of the process described above for the purposes of this thesis, is that this process involves a single product, with no regard to the other products in the family. The method to be developed and explained in subsequent chapters will consider single products as a part of a larger family, and will guide the selection of components for each member. It will naturally fit before the stage of production detailed design.

4.6 The need for product platforms

An interview was conducted with two long experienced senior engineers (29 and 34 years in industrial truck product development for the purpose of understanding the challenges associated with introducing a platform based product family in a real industrial setting. The following is a list of salient points:

- **Pros and cons**: *'Platform based products give a major advantage in logistic and financial terms, and also reduce development time and increase the capability to react to customer demand and offer the right product. On the*

other hand two products that are too similar may make one of them more difficult to sell, especially if the customers are able to upgrade a cheaper product themselves by changing a small number of components.' This quote shows the alignment between real experience and the pros and cons detailed in the literature review and awareness of the potential cannibalization issue such as the VW-Skoda issue reported by Shu (2005).

- **Traditional methods in use**: *'Decisions on whether to base a product line on a platform or what elements go in the platform are taken based on experience and ad-hoc analysis. A structured mathematical tool would be welcome but it would have to be validated and proven somehow before it is incorporated to the design process.'* Clearly there is an existing gap for comprehensive methods, but with an emphasis on reliability and confidence. Academic methods need to be validated in the real world in order to be embraced.

- **Coordination efforts**: *'Introducing a platform based product line is a major effort. It takes many resources from several departments.'* Another well documented issue in the literature (Pirmoradi et al., 2014), (ElMaraghy and AlGeddawy, 2014).

- **Legacy and transition from existing offer**: *'In the current range there are many individual parts designed for different product variants that could potentially be made common. This is due to them being designed independently with little communication between the different teams and the lack of a deliberate strategy to first design a platform and then the variants.'* This is the typical scenario for a bottom-up platform strategy as defined in the literature review chapter (Kalligeros et al., 2006), and has a profound effect as the benefits of introducing new common parts needs to be weighted against those of keeping the existing well-known parts.

- **Preferences**: *'For industrial vehicles, the main point that the customers look for is a low total cost of ownership - TCO -, which is not necessarily the option with the lowest purchasing price. However, often the total cost of ownership can be similar across different competitor products and then an appropriately differentiated product has its value.'* The total cost of ownership is hard to calculate as it involves many different costs and it is not easy to find an agreement on how every factor affects it. There are studies of TCO for passenger cars (Szumska et al., 2019), (Lebeau et al., 2013), but the case of industrial machinery such as industrial trucks is more complex due to the effect of the productivity that can be achieved with them being the main issue.

  It is also difficult to model the balance between costs and other factors that can make a product desirable, typically a company will define a set of requirements but there is a degree of arbitrariness to them.

- **Product variants requirements**: *'There are many engineering challenges in designing a common platform for several products. Perhaps the biggest one is to understand in detail the requirements of each variant, as they may be very different to each other. It is important to involve the experts in each model from the beginning.'* This quote highlights the importance of the front end issues and in particular the map from customer needs to functional requirements as identified by Jiao et al. (2007).

- **Performance trade-off and consistency across the range**: *'Unavoidably, there are conflicts between the different variants. Sometimes a decision that is the best for the whole range is not the best for a particular variant. In the early stages of designing a platform there are many uncertainties and sometimes engineers need to take decisions without detailed knowledge using educated guesses.'* This is an important point that highlights the uncertainties

inherent to the early planning and design stages where the platform structure needs to be defined.

It is clear that engineers are aware of the advantages and disadvantages of platform based design and it is infeasible in practice to offer a large range of products without some degree of commonality, as echoed by Otto et al. (2016). However, it is not so obvious where that commonality should lie for optimal results, and structured methods for that optimization process are not always well understood or even used. There is still work needed to align industrial practice with academic research (Simpson et al., 2001)(Simpson et al., 2014).

4.7 Generalization

The previous sections have introduced the world of industrial trucks, and this section attempts to generalize the problem to most other types of manufacturing industries. This problem of designing and manufacturing a range of similar but different products is common to many types of industries, such as automotive, domestic appliances, electronics, industrial machinery, computer equipment, etc. In general, designing and producing an individual product in a separate way for each possible application is expensive and inefficient, making it impractical (Simpson et al., 2001). When facing the design of a product family, there are a number of issues about requirements and implementation that need to be addressed. The list presented below is based on the case of industrial trucks but applicable to many other products. The issues are numbered arbitrarily for reference purposes, their number does not imply importance, chronology or order of any type.

Product architecture:

1. Identifying the applications for which the company want to offer a product.

2. Number of distinct products the company needs to cover all the identified applications.

3. Identifying what components can be shared among what products and what components need to be unique.

4. Balancing the advantages and disadvantages of component commonality.

5. Transition from the existing products to a platform based family.

Product performance:

1. Identifying and quantifying performance attributes to be considered.

2. Balancing some performance attributes with others.

3. Product performance consistency across the range.

The principal issues are, what do we want to obtain from our product family? what is the final objective? How can we say that a potential product family is preferable to other? All these issues are further elaborated in the next subsections.

4.7.1 Identifying the applications

To design a product family it is first necessary to know or define what kind of applications to cover. Each application can be defined by the intended use, the environment, and all the particularities that can make it different from another one. For example, an automotive family may be intended to offer city cars and family cars but not luxury cars. The applications can be a discrete set, e.g. a dishwasher and a washing machine, or a continuous range, such as electric motors from appliances ranging from a toothbrush to a home water pump. This is a top level definition of the problem and there is not a unique interpretation of the level of detail that distinguish one application from another. In the case of industrial trucks the borders between

different applications are blurred and some arbitrariness is required to separate them. For example, can it be considered as the same application two trucks handling the same loads but one in a 100 meter yard and other in a 200 m yard? The working cycle of the first one will include less driving as a percentage of the total energy consumed, but otherwise they are still similar applications that may be served with the same truck. This is also the case in industries such as automotive, but can be different for other industries where the limits between applications are much clearer, such as domestic appliances, where there is no middle ground between a dishwasher and a washing machine for example. They are clearly different applications, although there is still room for subtlety in distinguishing between a 7 kg and a 8 kg washing machine.

This is a front end issue (Jiao et al., 2007) named product family positioning or market segmentation (de Weck and Shu, 2003).

4.7.2 Number of distinct products

Another issue, similar but not identical, is how many products are required to fulfil all the determined applications. This question may appear in a similar way to the case of designing a product family from scratch, or in a different format when a family, or a product, already exists. In the latter case, the question will take the shape of *do we need an extra product to cover this new application or can we use an existing one?*, or *can we replace these two products by one or do we need to keep two?* Depending on how detailed the applications are, the number of products can be equal or lesser than the number of applications, since one product may cover more than one. Normally, the number of products is determined before any design work is undertaken, i.e. it is decided to design a product to cover an application, or a number of products to cover a number of applications. One of the purposes of this thesis is to provide a methodology to assess possible product family solutions

eliminating the hard constraint of the number of products. This is the concept of *product clustering* explained in the literature review chapter (Pirmoradi et al., 2011), and for the case of industrial trucks it is particularly important.

4.7.3 Identifying what components can be shared among what products

This is referred to as *platform identification* in the literature review (Chowdhury et al., 2011), and it is answered first in most methods, either one or two stage ones. This is a question directly applicable to every type of industry or family for which a platform strategy is even considered. Each product of a family is made up of a number of components. Some of those components can be common to several products, or even to all the products in the family, but what are the components that can be made common? and what are the specifications for those components so that they can be shared across several products? Those two questions cannot be considered independently, there are three possibilities, which will be illustrated using the particular case of industrial trucks:

- *A priori platform identification* The components to be made common are identified a priori and then the problem is what are the specifications that they must meet. e.g. We want to build a family of industrial trucks and decide that we will fit the same engine and pump to all of them. Then the task is to design or specify an engine and a pump that will be appropriate for all the products in the family. This approach requires a priori knowledge or experience to decide for what components this strategy is feasible. In the example given, the pump would not be a suitable part to be made common across the entire family, as this is dependent on the load to be handled. An experienced team could decide to share a pump between trucks from 1 and 3 tonnes, and another pump for trucks between 4 and 5 tonnes, for example. This approach

corresponds with either of the two diagrams in the top row of figure 2.4 and is a common approach across the industry.

- *Predetermined or preexisting components* A specific instance of a component with determined specifications is already known and selected, then the problem is to decide in what products it can be used. e.g. we want to build the family of trucks and we already have an existing engine in production, probably from the previous generation of trucks and we intend to keep using it. Can we use that engine for all the trucks? if not, for how many of them can we use it? We will need to find other alternatives for those products for which that engine is not suitable. This situation is common when an already existing family is extended, rather than designed from scratch. The scenario would be, we are adding this new product, can we fit the existing engine to it?

- *Open ended problem* The problem is open on both sides, there is a large number of ways to allocate component versions to product variants. How many product variants can share a particular component depends on what component version is chosen, and no a priori decisions are made. e.g. For the same family of trucks, we want to find the best way to distribute a yet undefined number of engine types among all the products. If we pick a particular engine, that will fit some of the trucks and we will need a second type for the rest of the trucks, or if we pick a different type of engine maybe it will fit all of them. This third type of problem is more complex, as it is often difficult to define what 'best way' means. This has to consider the wider picture of the whole family, not only that component.

These are three clearly different scenarios and are described for one component, a real case will have several components and may lie across several of these options. For example, we may be designing a family of agricultural tractors, which may have

an already existing engine to be used for at least some of them -case with pre-existing components- and we may be studying how many gearboxes are required for all of them -case of open ended platform-.

4.7.4 Balancing the advantages and disadvantages of commonality

Another issue directly applicable to every case that proves a difficult one to answer. It is specific to each problem and difficult to get agreement on. This is a trade-off by nature, and often each decision is taken on a case by case basis, depending on the importance conceded to it. Using the family of industrial trucks as an example, if we fit the same engine to a 3 tonnes and a 5 tonnes truck, the 3 tonnes truck may be overpowered, but the development will be easier, the purchasing price per engine cheaper and the service simpler. On the other hand, if we fit the optimum engine for each truck, the performance metrics are likely to be better, but at the expense of a longer and more expensive development, higher prices and more costly service. How can we compare both pros and cons?

4.7.5 Transition from the existing products to a platform based family

This question is applicable or not depending on the circumstances rather than the type of industry. The case in which some products already exist is common. In the real world, the case of a company creating a product family from scratch is rare although it may occasionally happen in cases of a major revamp or a hypothetical new company entering a well established market. Normally a company will already have an existing product line, or products and they would like to update them, substitute them or add new variants. The existing product line as well as the

order in which new product variants are added have an important influence on the problem, for example, the possibility of reusing an expensive piece of tooling may make a particular option more attractive than other, and even compensate for a potential poorer performance or more expensive parts. A platform that arises in this manner is referred to in the literature as a bottom-up platform (Kalligeros et al., 2006).

4.7.6 Identifying and quantifying performance attributes

This task corresponds to the front issue of mapping customer requirements to functional requirements (Jiao et al., 2007) and can range from a few for relatively simple products such as pens or coffee makers to dozens for complex machines such as industrial trucks or automobiles. Performance attributes are the metrics by which a product can be measured. They are part of the typical product specification sheet that the customers will read when deciding what product to buy. Although a product may have many performance metrics, not all of them will have the same importance, and finding the right balance between them is a problem on itself. A typical practice to design a product is to set a list of requirements and then chose a design that satisfies all of them. This is called 'satisficing' (Simon, 1956) and does not involve any process of optimization, it is common use in the industry as it is easy to understand and manage. However, this approach blocks the path to potentially more desirable solutions.

4.7.7 Balancing some performance attributes with others

The issue of balancing performance attributes is what makes a good product overall, how some features compensate for others. With several performance attributes to consider it is difficult to define clearly and unequivocally what makes a product superior to other. This is aggravated by the fact that some of the relevant perfor-

mance attributes are naturally in conflict with others, a typical example of this can be power and fuel consumption in a car. This leads to necessary trade-offs during the design, some attributes can only be improved at the expense of others, and all are further restricted by a target cost for the whole product. Most available methods do not clearly address this, although some do (Chowdhury et al., 2011).

4.7.8 Product performance consistency across the range

Performance consistency is important also for many types of industries, and related to brand reputation. Normally, a company will not be interested in designing a product family that includes good and poor products, unless they are clearly differentiated in ranges, such as basic and premium. In an extreme case, a company may prefer not to offer a product suitable for a particular application rather than offer one whose performance level is not consistent with the other products as this can be damaging for the brand and prevent customers from returning for other products.

4.7.9 What is the final objective?

Although the last on of the list, this is the main question that encompasses all the others when designing a product family. What is that the company intends to achieve with that family? There is not a categorical answer that fits all the cases. The objective is not simply selling the highest possible amount of products, or designing the best possible products. In general a company would want to maximize their profit, but may not want to do that at the expense of an enormous investment, or by offering products that are out of what they consider their niche. There are multiple factors and each case can be unique. Companies normally consider those factors in a list of key performance indicators (KPI) to measure their performance against their targets, but no universal KPI exist to measure the overall target for all the companies. Some of those factors are:

- Sales volume
- Gross profits
- Ratio income/outcome or margin of operations
- Reputation
- Geographic extension of the market

The 'best' product family is that one which maximizes the objectives of the company. How to define and model that is a complex problem in itself, it requires taking decisions, assessing factors and making assumptions with high levels of uncertainty.

4.8 Summary

The industry described in this chapter presents a problem in which a company has to offer a wide range of products with similarities among them but clearly differentiated requirements, similar in that way to other case studies in the existing literature. Most of the products share a similar architecture, i.e. a chassis, an engine or motor, transmission, wheels and a lifting mechanism. While at high level this is the same across the range, the parts can greatly differ for each product. This type of product families differs from cases such as that of electric motors, analysed multiple times in the literature and used almost as a de facto benchmark, in which the products are not only similar on a high level view but also component wise, being those components scalable and hence good candidates for continuous solving methodologies. Problems such as that one described in this chapter require a solving method valid for discrete systems, where there is no relation between two different choices for a component.

4.8.1 Case study approach justification

Figure 2.4 in the literature review chapter shows four different method approaches to solve the problem of designing a product platform-based range. After having described the particularities of the industrial trucks industry it is possible to assess which of those approaches is a better fit for this problem. The approaches are discussed in the same order in which they were described.

- *Two-stage a priori platform identification:* is a two-stage method and as such it requires to optimize or decide on the common parts first and the individual parts second. For the case of industrial vehicles there are parts that can be fixed as common a priori, before even knowing the specifications for that part, for example the seat, steering wheel and binnacle, or pedals. However, for other parts such as the engine, it is not easy to say, for example, that models from 2 to 5 tonnes will have one type of engine, models from 6 to 9 tonnes will other type of engine, and then specify those engines. This approach is unlikely to return a good solution unless it is based on prior knowledge. Even if this were not a problem, the first approach also requires optimization of those parts identified as common before having any knowledge of the parts specific for each model, and this is also flawed for this type of machinery where it is not possible to define the requirements for a component which is part of a chain independently of the other parts of the chain. For example, it would not make sense to optimize a pump for several trucks without knowing the specifics of the hydraulic system of each one of those trucks.

- *Single-stage a priori platform identification:* is a single stage approach and therefore the second problem of the previous approach is not applicable. However, the first problem is still relevant. Deciding on a priori common components can be done based on experience, but it relies on a belief that what has

been done before was ideal and will be in all circumstances, closing possible opportunities for exploration.

- *Single-stage no a priori platform identification:* is a single stage approach and the commonality architecture is not predetermined. This is a suitable candidate for this case as it is open to all combinations of common and individual parts and their specifications. However, it can still be improved by the next approach.

- *Single-stage no a priori platform identification non predetermined number of products:* In addition to the previous one, this also allows for different maps from applications to products, or different product clustering options, eliminating the need to take those decisions a priori. This is a considerable advantage for a case such as this one, where the number of applications is not clearly marked and there are many ways to cluster them into a manageable set.

The problem is now illustrated both theoretically and from an industrial point of view, the next chapter will explain the basics of the method that this thesis proposes and will claim as a contribution to knowledge in the field.

Chapter 5

Developing a method to design product family architecture

This chapter outlines and describes the method proposed to solve the identified gap at the end of the literature review. The method is original and the main contribution of this thesis.

5.1 Terminology used in this method

The following terms will be often used in this and the subsequent chapters. The definitions are specific for this thesis.

- *Component type:* each of the components which are part of a bigger machine in a general sense, comprising all the possible instances that can fulfil a function, e.g. an engine.
- *Component instance:* each particular component that can be used to fulfil a specific function, e.g. a particular model of 2.5 litre Diesel engine from a defined supplier.
- *Pool of components:* set of all the available component instances.

- *Product candidate:* each possible combination of component instances to create a product that fulfils a particular application.

- *Pool of product candidates:* set of all the possible product candidates for all the applications to be fulfilled. This set is divided into as many subsets as applications exist.

- *Family candidate:* each possible combination of elements from the pool of product candidates formed by selecting one element from each application subset.

- *Pool of family candidates:* set of all the possible family candidates.

- *Method:* the whole process that is described in this chapter and demonstrated in the next one.

- *Step:* each of the sequential tasks to be followed to accomplish the method.

- *Performance model:* model that maps a product candidate to the figures it would achieve in the different performance attributes under consideration.

- *Attribute goodness:* value between 0 and 1 that relates a particular performance figure to its desirability for a particular application.

- *Aggregated goodness:* value between 0 and 1 that relates a product candidate to its desirability for a particular application.

- *Objective model:* model that provides a score for each family candidate against one or several objectives.

- *Search:* or searching algorithm, is the step of the method that searches for the solutions.

5.2 Principles for the method

This chapter will present a method to aid the decisions on the structure of component commonality across a product family. The scope is comprehensive from the applications to be covered to the structure of platform and variants and the characteristics of each component, and it adopts a single stage approach. The method proposed will provide a Pareto set of solutions with conflicting objectives, such as, for example, performance and costs as independent variables, leaving the decision maker with a visual envelope of possibilities to analyse before deciding on the final option. The novelty resides in three main points:

- Product variants are not predetermined. They are one of the outputs from the method.
- Performance optimization and aggregation of performance is done with a fuzzification model, requirements are given as fuzzy sets instead of a list of hard constraints.
- Truly multi-objective approach from intended applications to product family structure and variants solved with a multi-objective genetic algorithm.

These three points are further explained in the next subsections.

5.2.1 Product variants not predetermined

Normally, the product clustering and family design problems are treated separately, first and second problems. This allows for having two more contained problems at the expense of optimizing the two of them independently, with the implied risk of missing solutions. For the case of industrial trucks, this clustering is typically done in two levels, the first of them is the fundamental type of application for which the truck is intended, such as towing, general lifting and carrying, or warehouse; and

the second level is a division in load capacities. This thesis will propose a method to consider the applications to be covered rather than the products to cover them, so avoiding the constraints generated by the solution of the first problem and typically passed on to the second problem. This generalization increases the computational complexity, and is not feasible for any arbitrarily large number of applications. In cases where the number of applications is too large, a reduction may be necessary.

5.2.2 Performance optimization

The method presented in this thesis does not consider a list of requirements for each product, but the degree of how good a particular performance attribute is for each identified application. This approach allows for better products to have preference over others which just meet minimum or desired requirements.

5.2.3 Multi-objective approach

A multi-objective treatment of the problem presents two main advantages over a single objective treatment:

- Balancing several objectives into one is difficult to do and to check that the combined function truly reflects the preferences. In the case of industrial trucks there are many performance attributes that are important for the customer and it is very hard to define a function that combines all of them together with the purchasing price to provide a score that really reflects the preferences and can be used to categorically say truck A is better than truck B. It is necessary to consider all the objectives. A multi-objective approach shows the set of solutions for all kind of balance between the objectives. This is particularly true with two objectives, as the set can be shown in a 2D plot easy to visualize. For higher numbers of objectives the visualization is less intuitive and can become overloading for the person analysing it.

- Performing trials with different ways of defining a single preference function is computationally expensive, since it is basically repeating the entire problem again. A multi-objective approach provides this analysis for any articulation of preference in one go, although this run will typically take longer than a single instance of single objective.

5.3 Method overview

The problem that this method is addressing can be posed as: *Find the best product families and commonality strategy to fulfil a number of applications from an existing or hypothetical pool of components without imposing restrictions by a priori decisions on commonality or product clustering.*

Those undue restrictions, or a subset of them, are commonly found in several methods available in the literature, and they refer to:

- Number and characteristics of variants predetermined.
- Platform and variant specific components predetermined.
- Variables of a continuous nature only.
- Performance defined as a simple attribute easy to estimate.
- Objectives based only on performance or wide statements such as sharing the maximum number of components. No balance between pros and cons of a platform approach.

The problem can be broken down into several sub-problems or questions. From the statement of the problem, it is clear that the solution will take the shape of a product family, i.e. a number of products that will cover a number of applications.

So, **what are those applications?** And, **what are the performance attributes** that need to be considered?

Furthermore, the products in the family will be built with some common and non common components selected from a pool of existing or hypothetical components, **what is that pool of components?**.

Defining a family from a pool of components is a combinatorial problem, and there can be a large number of possible solutions, but the problem asks for the best. Best is a term that requires some criteria to define it, **what does best mean?**, and **how can we find it or them?**.

It is obvious that in order to find the best family according to some criteria it is first necessary to be able to assess a family against those criteria, **how can we know how good is any given product family without building it?**

The next subsections will look into these questions in more detail.

5.3.1 What are those applications?

This question consists in identifying the applications for which the company intends to offer a product. Each application is an intended use that can be differentiated from the rest. As it has often been mentioned in this thesis, *application* is not synonymous with *product*, an application is a use whereas a product is a particular set of components intended to fulfil at least one application. There is a map between applications and products that is not necessarily one to one. An example of that map, for illustrative purposes only, is shown in figure 5.1. Ultimately, there is a map between applications and components, with products as a step in the middle. The method presented in this thesis assumes the applications at the top and the components at the bottom, and generates the products in the middle that map the two extremes.

The set of product applications, once identified, is a fixed set for the remainder

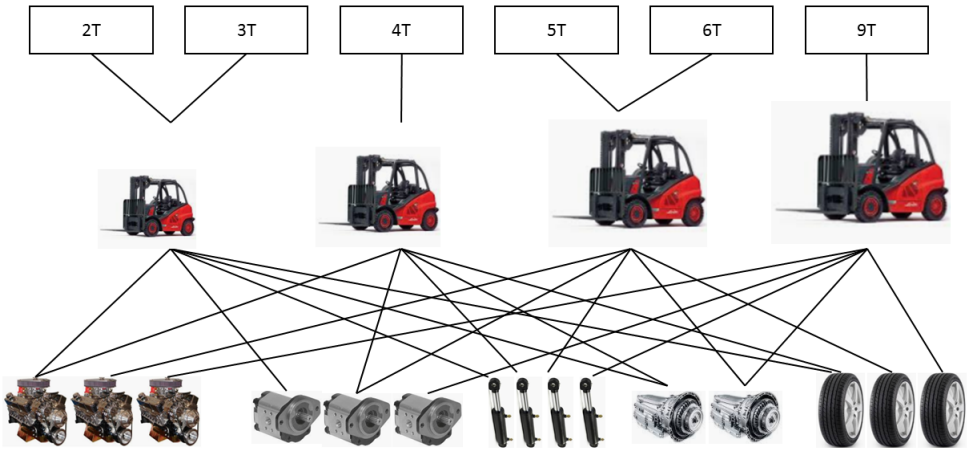


Figure 5.1: Applications to products map

of the problem resolution, and an input for the next steps, whereas the map between applications and products can and is expected to vary for different potential solutions. It is also important to clarify the scope of this thesis regarding the definitions of the applications. This step is part of the described method, but it is taken as given information, the methods to define the target applications are not part of this project.

5.3.2 What are the performance attributes?

Once the applications are determined, it is necessary to identify the attributes against which a product will be measured. They are the important features that will define whether a potential product can fulfil an application and how well it does so.

5.3.3 What is that hypothetical pool of components?

At this point, the possible components that will make up the products need to be identified. The components need to be characterized so the different performance

attributes can be modelled as a function of them.

As an example to provide an idea of how the combinatorial numbers escalate, if the problem involves a family to fulfil 5 applications, each application requires 6 component types, and there are 7 component instances for each component type, then each application can be implemented in $7^6 = 117649$ ways, the pool of product candidates will have $5 * 117649 = 588245$ elements in 5 subsets of 117649 elements, and the pool of family candidates will have $117649^5 = 2.25 * 10^{25}$ elements. This example shows how the complexity of the problem increases from a manageable number of candidates for a particular application to a very large one when the whole family is taken into account.

5.3.4 How can we know how good is any given product family without building it?

This project shows a method that ultimately balances performance with costs at a time when actual products do not yet exist and performance tests cannot be conducted. For that assessment to be feasible, a method is required to know the performance, or at least an estimation, for all possible designs under consideration. This exercise involves one way or another of modelling, and in particular, mathematical modelling for reasons explained in the Literature review. The models can range from simple extrapolations from known products to general equations or full models simulating the product in its intended use. The simplest modelling technique would be extrapolating from previous knowledge, but this is not always reliable for complex machines, as their performance depends on a chain of components that interact with each other in a non-linear way. As an example to illustrate this point, the length of the winding can be used to calculate the torque in an electric motor, and there is a clearly defined relation between the torque of two motors that differ only in the winding length . However, the case for machines such as industrial trucks

is different. If a particular truck with a particular engine and a particular gearbox yields a fuel consumption of 4 l/h, and it raises to 5 l/h with a second gearbox, this represents an increase of 25%. If a different truck with the same engine and gearbox consumes 6 l/h, and we replace the gearbox with the second gearbox, a simple extrapolation would predict a new fuel consumption of 7.5 l/h, but the actual fuel consumption may be very different, or even move in the opposite direction in the case that the second gearbox be the optimal for this second truck. Also, in some cases, simpler equations can be used as surrogate models, or metamodels, based on the performance of known machines to fit the existing data (Collopy and Hollingsworth, 2009). The most common techniques in that field are Kriging and Response Surface Methodology, which is a polynomial approach (Kleijnen, 2017). However, those metamodels become harder to define as the number of inputs or dimensions becomes larger, not only due to the increasing mathematical complexity, but also because in a complex machine there are many discontinuities that are hard to spot a priori or with a superficial analysis. For example, in a lifting system in a truck, the lifting speed may seem to follow a curve depending on the pump displacement, but at one particular displacement the system may not get enough pressure to overcome the friction and not be able to lift at all. It is difficult to know that point without a thorough analysis, and this effect is further compounded by the fact that this point can be different when a different engine is fitted. This example serves to illustrate that metamodels can not be relied upon when the real problem involves complex machines with many variables.

5.3.5 What does *best* mean?

Best is a very loosely defined term, but for the purposes of this thesis, it can be considered as **that solution that better fits the preferences of the decision maker**. For problems with a unique objective, best is the solution that maximizes

or minimizes that objective, but when more than one objective exists, as it is the case with most engineering problems, it is generally impossible to obtain a solution that maximizes or minimizes all the objectives at the same time. In those cases, best refers to the solution that better fits the compromise or balance that the decision maker assigns to the different objectives. As shown in the Literature review, this can be done by defining an a priori criteria for that compromise, or a posteriori by choosing among a set of possible solutions that are part of, or as close as possible to, the Pareto set. There is no universal method to define *best* for multi-objective problems. Defining an a-priori preference function is not an easy task, and it will obscure all the potentially good solutions bar one from the decision maker's sight. On the other hand, when the number of objectives becomes too high, the set of solutions presented to the decision maker for assessment also becomes very large and with no apparent order, which is not helpful. This method incorporates a compromise approach by combining the objectives to present a manageable number of them, which normally means no more than three and preferably two. The method recommendation is to combine the objectives in a way that is meaningful to a decision maker, such as pros and cons, cost and performance, etc. Simpson (2004) identified three assumptions common to multi-objective approaches:

- Maximizing performance of each product increases their demand. However, it is in general not possible to maximize all the performance objectives of all the products at the same time, this is a balancing problem in itself.

- Maximizing commonality across products minimizes production costs. While this is often true, there are cases in which it does not hold. Increasing commonality may require using more expensive parts resulting in higher overall costs.

- The trade-off between the previous two provides the most profitable family.

The next chapter will show an example of that trade-off by combining several performance objectives on one side, and considering the cost of all the components - rather than commonality on its own - on the other side.

5.3.6 How can we find the best one?

As it has been explained in the section on the pool of components, the pool of family candidates, or possible solutions is normally very large, and it is not possible to evaluate all of them. It is necessary to use some kind of searching algorithm, there are several methods available and they have already been mentioned in the Literature review. The approach of this thesis is to treat the problem of optimizing a product family as a multi-objective optimization problem, i.e. generating a Pareto front of solutions from which the decision maker will be able to pick the preferred one. This is an a posteriori articulation of preferences case, as opposed to a priori articulation of preferences where the decision maker preferences are modelled into a function, turning the problem into an effectively single objective one. Due to the characteristics of the problem, as it will become clear with the case study, including discontinuities and discrete components, the problem is best tackled with evolutionary algorithms, and the chosen one for the case study is a genetic algorithm. This choice is not compulsory or an integral part of the whole method described in this chapter and no novelty can be attributed to the searching algorithm per se. This is only an example of how suitable solutions can be found for the problem of designing a product family. For different case studies the particularities of the searching algorithm may vary.

5.4 Ordering the steps - Method description

The answers to the questions posed in the previous section will be worked following a logical order, and this will result in the description of the method proposed in this thesis. This method follows the right bottom corner diagram of figure 2.4, i.e. single stage, non a priori platform identification, non a priori map from applications to products. It is divided in the following main steps:

1. Identification of product applications
2. Definition of performance for each application
3. Definition of the design pool
4. Product simulation to find performance
5. Definition of objectives
6. Optimization

Figure 5.2 shows how each step is geared towards the questions in the previous section, whereas figure 5.3 shows another diagram where the steps are mapped to a reprint of the one shown in figure 2.4 in the literature review that illustrates the objective of this thesis and. That diagram in figure 2.4 started with the applications that need to be covered, and through an optimization process, it outputs the different product variants, as well as what components are part of the platforms or subplatforms and what components are individual for each variant. The diagram now in figure 5.3 shows a one to one relation for the identification of the applications, with the other steps of the method being a break down of the previously termed *optimization* in general. The dashed lines on the right hand side represent possible iterations and optimization re-runs if the found solutions are not satisfactory. Whereas those on the left hand side represent how those iterations would work

in practice, by either redefining the objectives because they do not represent the preferences adequately, or there needs to be a rethink of the potential components. The main purpose of each step, discussion, and how they can be accomplished in general is detailed in the next sections. The following chapter will then go through the steps as they were implemented in the particular case study used in this thesis. This process, and how each step is implemented, is original and it is the main contribution that this thesis claims.

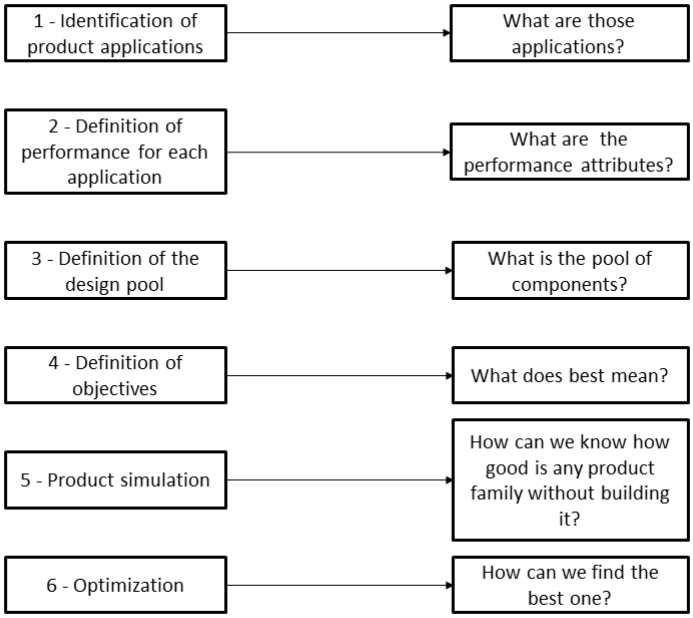


Figure 5.2: Map between steps and questions

5.4.1 Identification of product applications

The applications that the product family is intended to cover is taken as given information, and as mentioned before, this method does not provide guidance on how to accomplish this. There are many ways to separate and classify the amount of products intended to be sold in a discrete number of applications, ranging from an ideal, and unmanageable, one to one products unit to applications map, to just a few

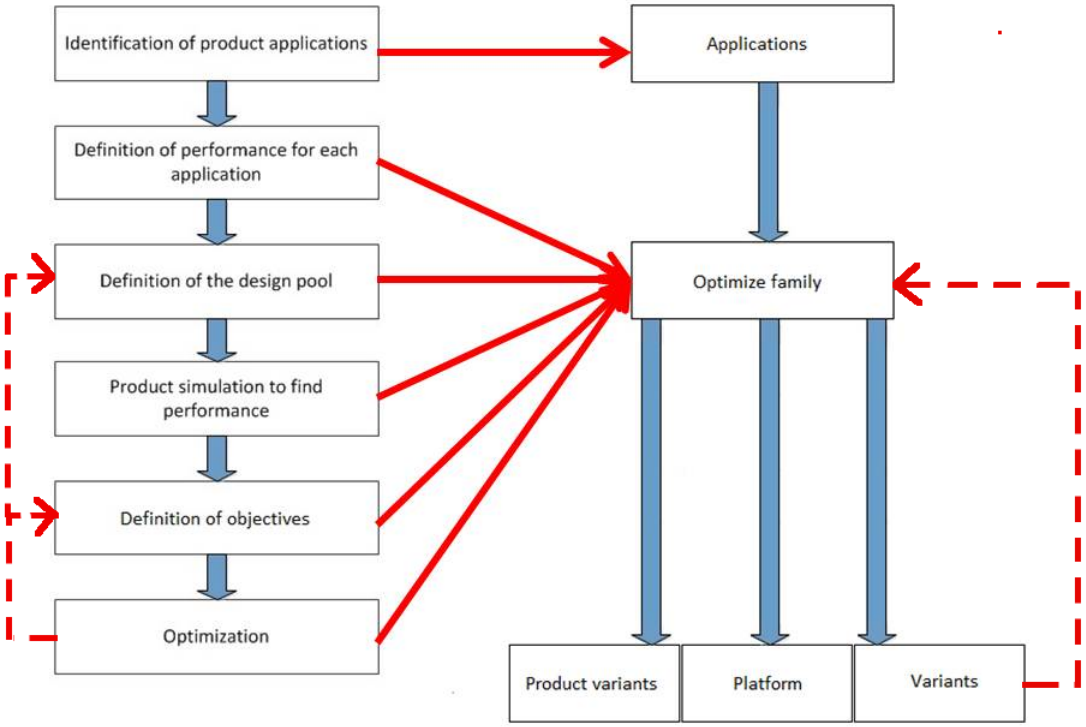


Figure 5.3: Map showing the main steps

differentiated applications that wrap all the product units to be sold. The complexity and required work for the remaining steps of this method is heavily dependent on that. As a general rule, the method will consider the set of applications as it is normally done in the industry or company where the family lies.

5.4.2 Definition of performance for each application

Once the applications are identified, it is necessary to define the most relevant performance attributes for each application and quantify them. At this point, only strictly performance attributes of the final products are considered, disregarding any manufacturing, design or procurement issue. These attributes do not need to be the same for all applications, and less so the quantification. The recommended method to define the performance attributes is with a panel of people chosen from engineering, marketing and product management who understand the intended product positioning in the market, the requirements, and the competitors state of the art. As suggested, this phase can be further split into two sub-phases; identification and quantification.

Identification of performance attributes

The different target applications are separated and an analysis is performed for each one. The identified attributes need to be the key characteristics that will distinguish a particular product from the competitors, the attributes that the customers will look for and will use to decide which of the available products they will purchase.

With the exception of very simple items, most products in general can be characterized by many performance attributes. As an example, a car is normally issued with a long list of specifications and performance, with attributes including acceleration, maximum power, torque, urban fuel consumption, highway fuel consumption, boot capacity, head room, etc. The list spans dozens of items. Even an apparently

simpler machine, such as a washing machine, can still be measured against several performance attributes: speed, energy consumption, water consumption, mass of load per laundry, number of programs, etc. Each performance attribute depends on a particular set of components, while other components may not have any effect on it. For example, the fuel consumption of a car depends on the engine, vehicle mass, gearbox, tyres, aerodynamic shape, etc. but it is independent of the material choice for the seats. When the team is designing the seats, they do not need to consider the effects it will have in the fuel consumption, and when the team is designing the power train they do not need to consider the seats, as they are independent of each other. Hence, only the attributes that depend on the components being designed need to be considered.

Additionally, the performance attributes may not be the same for different applications. For example, when designing a family of domestic appliances, the attributes that need to be considered for a washing machine will differ from those for a dishwasher, even though the two products can still share several components such as the body frame, motors (although highly unlikely) or electronics.

Considering individually all the performance attributes for all the products of a family will result in a large set of objectives, and as it has been shown in the Literature review, this would make the final set of solutions nearly as big as the pool of family candidates, which would not add any value to solve the problem. In order to keep the problem tractable and being able to offer a solution, it is normally necessary to combine some of the objectives.

Fuzzification

To facilitate the combination or aggregation of difference performance attributes, which will be necessary at least to some degree to define the problem objectives, each different performance attribute is measured in units of goodness. The nature

of these units is intuitive, i.e. how good the figure for each particular performance attribute is. Goodness ranges from zero to one, where zero means that a particular performance figure is not acceptable, and one means that it is the best that it can be, impossible to improve upon. A function, or fuzzy set is defined for each attribute to convert a performance figure to goodness. The simplest way to obtain that function is by defining several membership levels, for example:

- 0: Unacceptable value. Any product with a single performance attribute with a membership of zero will be discarded as invalid.
- 0.05: Just acceptable, barely enough to prevent the product from being directly discarded.
- 0.2: Passable, performance would be slightly disappointing but still acceptable.
- 0.5: Fair, what they would expect to have in the final product.
- 0.8: Good, this performance would place the product among the top competitors.
- 0.95: Very good, market leader in that particular attribute.
- 1: Ideal, Nothing could be better than this. This can be either an unachievable point, such as a fuel consumption of 0 litres per hour for a vehicle, or a point beyond which there is no real benefit, such as a legal driving speed for a vehicle limited by regulation.

Then the panel selected to provide their input to build the membership sets are asked to provide a value for each performance attribute that they will consider passable, fair, good, etc. and the value that they will consider as the bare minimum that should be achieved. This exercise will generate a more or less populated cloud of points such as the one shown in figure 5.4. The set, red line in the figure, is

then defined based on those points. The shape of the function defines the relative importance of improvements depending on the area in which they happen. An area where the curve is very steep means that small improvements of performance make a considerable difference, whereas an area where the curve is nearly flat means the opposite. There are multiple methods to draw the set, such as calculating the average for each level, piecewise linear regression, splines, etc. But an important point is that, for monotonically increasing sets such as the one in this example, the extreme membership levels 0 and 1 should coincide with the maximum of the unacceptable level and ideal level, to reflect the most stringent views from the panel. In the case of a monotonically decreasing function, an attribute for which a lower value is better, those levels should be taken as the minimum of the responses rather than the maximum.

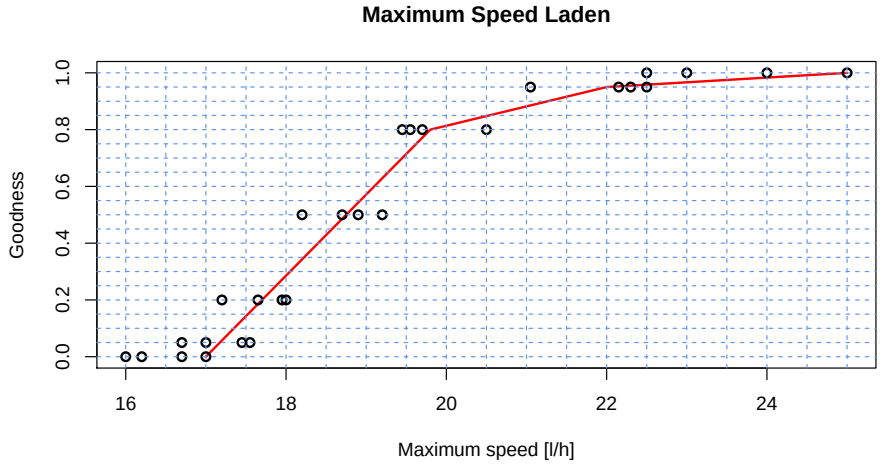


Figure 5.4: Example of an attribute fuzzy set

Once the sets are defined, it is possible to assign a level to any value of that attribute, either by interpolation or application of a curve function, depending on the method used to define the set.

Aggregation of performance

Normally, there are several performance attributes identified for each product, and it is not manageable to keep every one as an objective for the optimization process. It is convenient to define a single performance measure for each of the products in the family that aggregates all of its identified attributes. In this thesis, that aggregated performance measure is called *product goodness* and there are multiple existing methods to define it as mentioned in the literature review, such as weighted means, Tchebycheff, lexicographic, etc. What is important is that the method accurately reflects the relative importance of each attribute and is capable of ordering different products in a meaningful way with the agreement of the team that defines it.

An example of performance aggregation method, and the one that will be used in the case study, is a weighted sum with all the attributes and an exponent to define the level at which some good attributes can compensate for poor ones (Dai and Scott, 2006).

$$G = \left[\frac{\sum_{i=1}^{9} w_i * Pg_i^s}{\sum_{i=1}^{9} w_i} \right]^{1/s} \tag{5.1}$$

Where G is the aggregated goodness for a given product candidate, w is the relative weighting and Pg is the performance goodness on each attribute. The parameter s is used to vary the level of compensation between different attributes. A value of $s = 1$ results in the well known weighted arithmetic mean, whereas values of $s < 1$ give more relative importance to low performance attributes, i.e. low degree of compensation, and values of $s > 1$ give more relative importance to high performance attributes, i.e. allow for a higher degree of compensation between the different attributes.

In addition, the number obtained from that formula is multiplied by a function

C that is defined as follows:

$$C = \begin{cases} 0 & if P_i = 0,\ for\ any\ i \\ 1 & otherwise \end{cases} \tag{5.2}$$

This ensures that a variant with an unacceptable attribute is spotted and discarded.

The different weight parameters of each performance attribute for each application and the compensation parameter must be drawn from interviews with people with expert knowledge who can assess the relative importance of each attribute.

5.4.3 Definition of the design pool

The design pool is the set of all the possible components that can go into the products and are relevant for the study of how to architect the product family. This set must include the components that may or may not be made common across the product family or part of it, and that will have an influence on the different performance attributes under consideration.

The characteristics of the components of the pool can typically be obtained from data sheets when those components are sourced from existing suppliers' offers, or conceptualized, in the case that the components are yet to be designed. All potential component instances must be included in the pool.

5.4.4 Product simulation to find performance

This step consists of designing the necessary models to map a choice of components for each product application to a set of attributes, i.e. calculate or estimate quantitatively what the different performance attributes of a product designed with those components will be. This is shown in figure 5.5

The simulations need to be accurate enough to produce reliable results, but at the same time they need to be run reasonably fast, as the searching algorithm will

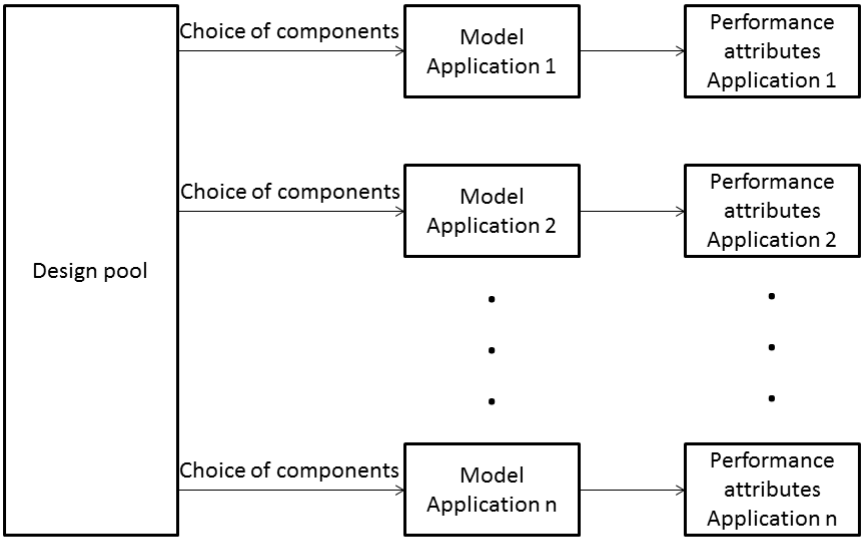


Figure 5.5: Simulation models map the choice of components for the n products that compose the family to n sets of performance attributes

need to test many combinations of components for each application, typically in the hundreds, thousands, or higher. For this reason, simulations based on full size finite elements models, CFD, or similar may not be appropriate for most cases. If this type of modelling is relevant to the problem, then simplifications or conversion to tables are recommended.

5.4.5 Definition of objectives

This step is related to the question *what does best mean?*. This is the definition of what is expected from the family and the products that compose it. In general there are many objectives, both for each individual product and for the family as a whole. Some typical objectives are the performance attributes that will be calculated with the simulation models, but there are other types of objectives such as costs, development time, commonality, or even hard attributes to measure such as, for example, market disruptiveness or saleability. The selection of the objectives

requires an exercise with technical and commercial input. Generally there are three tasks involved:

1. Objective identification: Identify all the relevant objectives for each product and the family as a whole. This should include the performance attributes that were selected in the previous steps, and also all the costs that depend on the choice of components, such as the costs to source them.

2. Objective quantification: This consists of assigning priority or importance to each one of the identified objectives.

3. Objective aggregation: Combine or aggregate all the objectives previously identified into a shorter and manageable set. For the problem to provide solutions that can be meaningful, it is recommended to limit the final number of objectives to two or three. Several combination techniques are available, such as weighted sums, best minimax, etc.

Even if the problem consisted only in optimizing the performance of a product family, without giving consideration to anything else, the number of performance objectives would typically make for an intractable problem, in which nearly every potential family would be a Pareto solution. It is therefore necessary to compromise and reduce the number of objectives by combining or aggregating some or all of them. The approach chosen for this thesis is to have a reduced number of main objectives to optimize, normally two or three. In case of two, one of them would represent the goodness, attractiveness, or customer appeal of the product family irrespectively of the cost, and the second one the costs associated to obtain that family. In case of defining three final objectives the third one can incorporate other features such as development time or flexibility to add new products when necessary. The first of the objectives needs to be the aggregation of all the performance objectives identified as relevant, and for the case study shown in the next chapter it will follow a process

shown in figure 5.6. Each of the single performance objectives of each variant is converted into a unit of goodness ranging from 0 to 1, where 0 is a performance figure that is unacceptable, and 1 is a performance figure that is as good as it can be, or ideal. Those performance goodness figures are then aggregated using the preferred method which is problem specific. In the case study the method of aggregation will be a weighted mean allowing for compensation to obtain a single goodness for each of the products that will fulfil the identified applications. The last step is to aggregate the single score of each application. A possible method for that, and the one that will be used in the case study and shown in chapter 6 is to estimate the sales and price for each application depending on its aggregated score and add the numbers for the entire family. It is important to consider family candidates only if all the attributes for all the applications are at least acceptable according to the defined criteria, otherwise the particular product with an unacceptable attribute would not saleable and the family candidate would be lacking a product to cover that application, deeming it invalid.

For the objective that considers the costs this thesis proposes, and will show in detail in the next chapter, to add the cost of all the components necessary to build each family candidate taking into account the number of units intended to be manufactured and the variation in price that can be expected depending on the purchased volume.

5.4.6 Optimization

The searching algorithm runs to find the best combinations taken from the design pool. Due to the characteristics of this type of problems, in general evolutionary algorithms are a good choice to solve them. Qualitatively, the searching algorithm will run through a number of iterations evaluating the fitness function, i.e. calculating the objectives for one or several potential product families and choose those

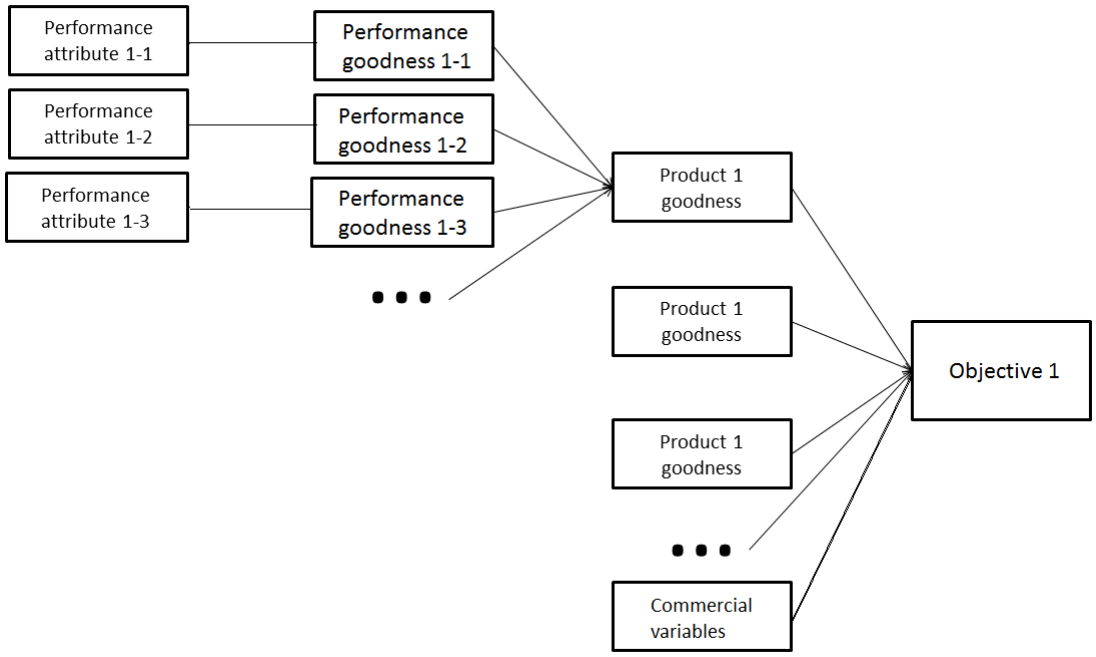


Figure 5.6: Diagram showing the performance aggregation layout

with the highest scores. For each iteration it selects a new set of potential families based on the history, with the intention of finding better candidates than the current best. The case study presented in the next chapter will use a multi-objective genetic algorithm that incorporates a modification to run only part of the fitness function in each iteration. Part of that function will be run as a pre-process before the start of the first iteration. This will be explained in more detail in the next chapter.

5.5 Summary

This chapter has outlined the method and its phases in a general way. Figure 5.7 shows the techniques that are relevant for each of the steps that compose the method. None of the techniques is new on its own, the novelty resides in the integration to solve the problem of designing a product family. The next chapter will show how it works over a case study, going deeper into the details.

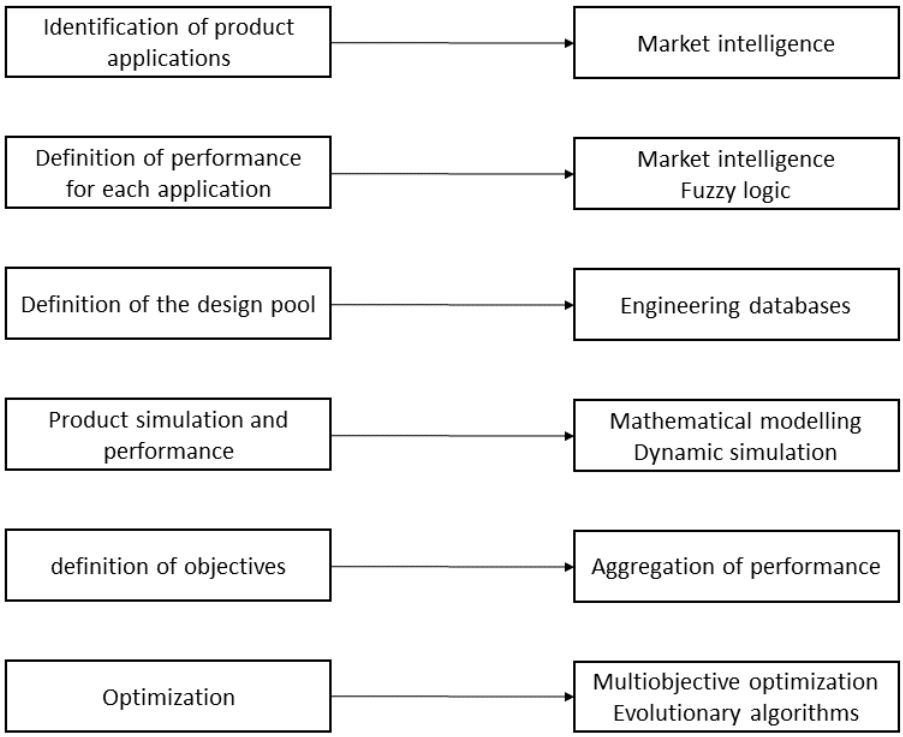


Figure 5.7: Techniques used for each step

Chapter 6

Case study

The previous chapter described the method proposed in this thesis in a general way. This chapter presents a case study and will go through the different steps detailing how they can be implemented. The case study is the design of a family of counterbalanced forklift trucks powered by Diesel engines and intended for applications ranging from 2 to 9 tonnes.

6.1 Identification of product applications

In this case study, the applications are defined by the load to be lifted and carried. This is a natural decision as it is the main difference between the existing products in all brands. There is a set of 12 different applications: 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 7, 8 and 9 tonnes.

6.2 Definition of performance for each application

This step consists on understanding the product requirements to fulfil each application and identifying the performance attributes that are the most relevant for the products under consideration. This step requires deep knowledge of the applica-

tions and what is required to fulfil them. For the case study, this knowledge was acquired based on the author's experience and interviews with both engineering and marketing experts. Nine performance attributes were selected, they are:

- *Fuel consumption:* one of the main contributors to the total cost of ownership of a truck, and hence one of the most important figures.
- *Maximum speed:* directly related to the productivity of a truck, specially for applications in big yards with driving focused working cycles.
- *Lifting speed:* also related to the productivity, although this feature is more relevant to warehouse equipment, it still applies to any type of working cycle.
- *Drawbar pull:* is the force that the truck can apply at a given speed of 1.6 km/h, or 1 mph. Although towing is not the main use for trucks, it provides a good reference for their capabilities and how it will be affected by the loads carried.
- *Gradeability:* is the slope that the truck is capable of climbing steadily at the same given speed of 1.6 km/h. More or less relevant depending on the intended use, but ramps are common in many yards and loading docks.

With the exception of fuel consumption, all the other items need to be provided both with an unladen truck and carrying the rated load.

6.2.1 Identification of performance attributes

For the case study, there is a total number of performance objectives of $12 * 9 = 108$. Those performance objectives are measured in physical units and then converted to goodness units through a fuzzification process as explained in the previous chapter. Then all the scores relevant for each application are aggregated into a single measure

of goodness using a weighted sum that considers the relative importance of each attribute and the degree of compensation that is allowed.

6.2.2 Fuzzification

The conversion from physical units to goodness units was done by defining fuzzy membership sets for each performance attribute and then calculate the membership of each achieved value. The process to define those sets consisted on holding discussions with personnel from Engineering and Sales & Marketing to provide ratings for what they would consider unacceptable, acceptable, good, etc. values for the different performance attributes for each application.

The information received was codified to define a fuzzy membership set for each attribute. The fuzzy sets were defined with 7 levels of membership as described in chapter 5. A total of 108 fuzzy membership sets were constructed based on the given ratings, they are shown in figures 6.1 to 6.9. There is one plot for each performance attribute and the twelve applications are represented in each plot. In cases where only one line can be seen, such as gradeability, is because the desirability of each value is common for all applications as identified by the personnel involved in the study.

Fuel consumption in figure 6.1 shows a unique curve for each application because the expectations cannot be the same for different loads. In this case, a goodness of 1 is unachievable, corresponding to a null consumption, as the lesser fuel consumed is always the better without any limit beyond which a reduction would not represent an advantage. The curves take goodness of 0.95 for values that would be top of the market and falls to zero for values that would be uncompetitive. The drop is not linear, reflecting a bigger influence at high levels of goodness, i.e. small improvements on an already good fuel consumption provide huge market advantages.

Travel speed in figures 6.2 and 6.3 show the same curves for every application,

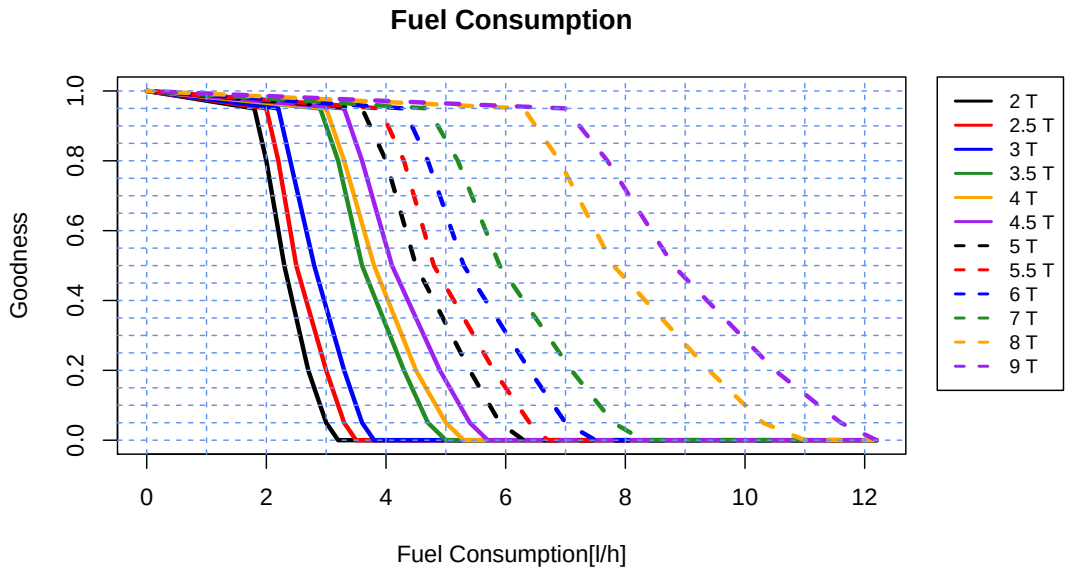


Figure 6.1: Fuel consumption fuzzy membership set

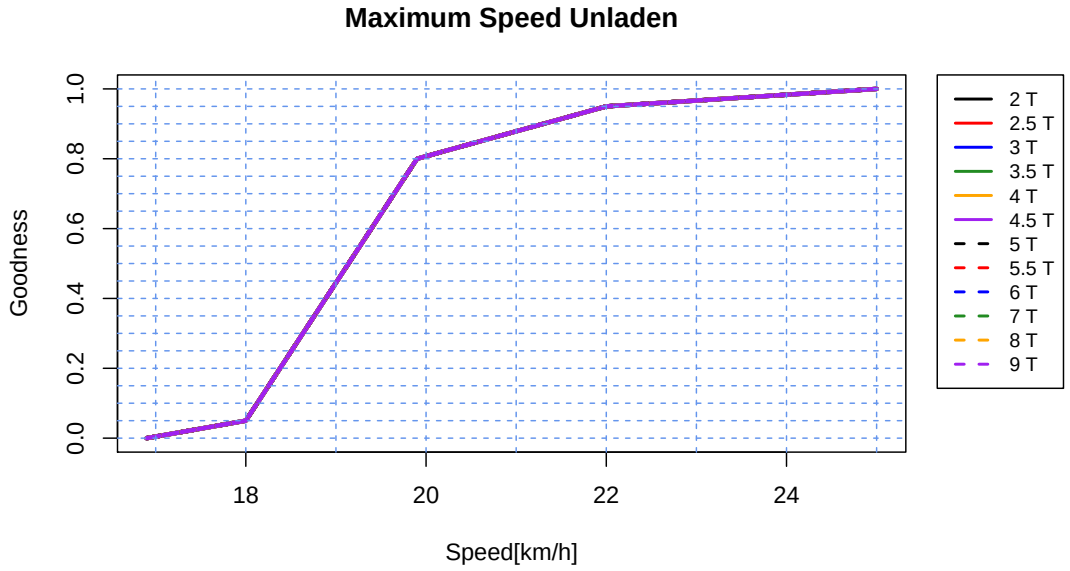


Figure 6.2: Maximum speed unladen fuzzy membership set

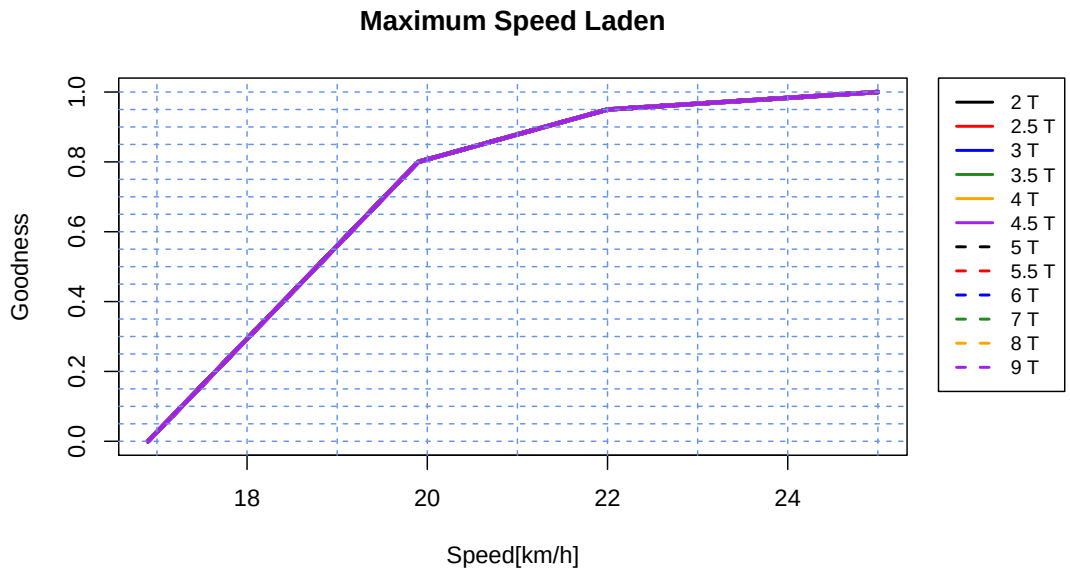


Figure 6.3: Maximum speed laden fuzzy membership set

with two clearly defined extreme points, 17 km/h a minimum acceptable top speed and 25 km/h as the speed not worth exceeding. The curve for a laden truck goes above the unladen one, as a given speed has more merit when achieved with the truck laden.

Lifting speed in figures 6.4 and 6.5 show similar characteristics as travel speed, although in this case the curves are displaced to the left for higher capacities. This is due to customers expectations namely higher for lighter trucks.

Drawbar pull in figures 6.6 and 6.7 are the attributes most clearly distinguished for each application, as it is a scalable characteristic, a heavier truck requires a higher drawbar pull. In this case, the curves are closer to linear between the unacceptable values and those that would place the trucks at the top. The reason is that this is not an attribute that makes a huge contribution to the overall goodness, and it is not easy nor worth it to differentiate between the regions of the curve.

Gradeability in figures 6.8 and 6.9, like drawbar pull, is not the biggest contributor to overall goodness and the curves are also less detailed. The difference is that this is not a scalable attribute, and hence the curves are equal for all the

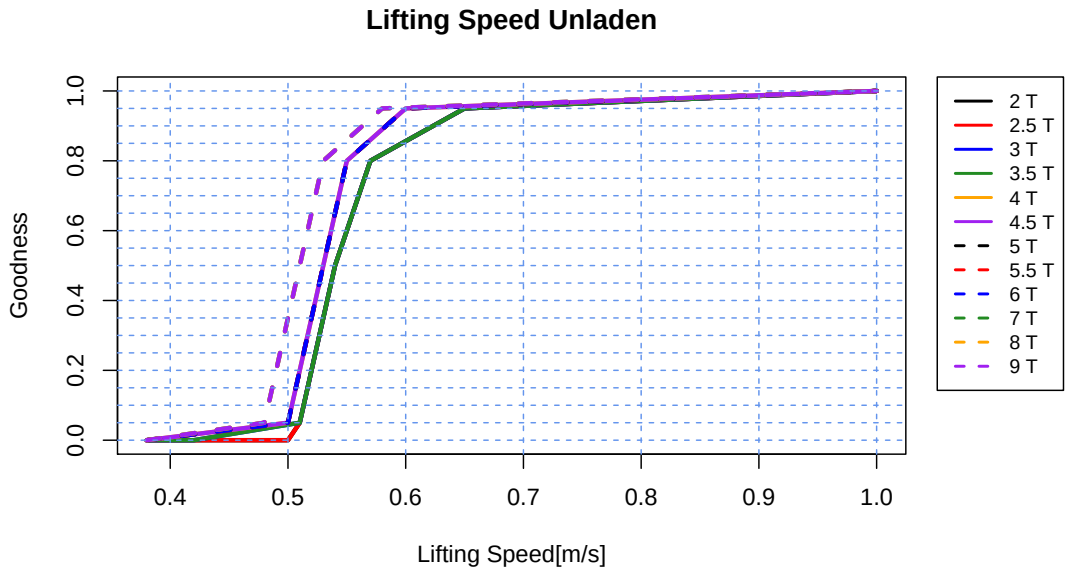


Figure 6.4: Lifting speed unladen fuzzy membership set

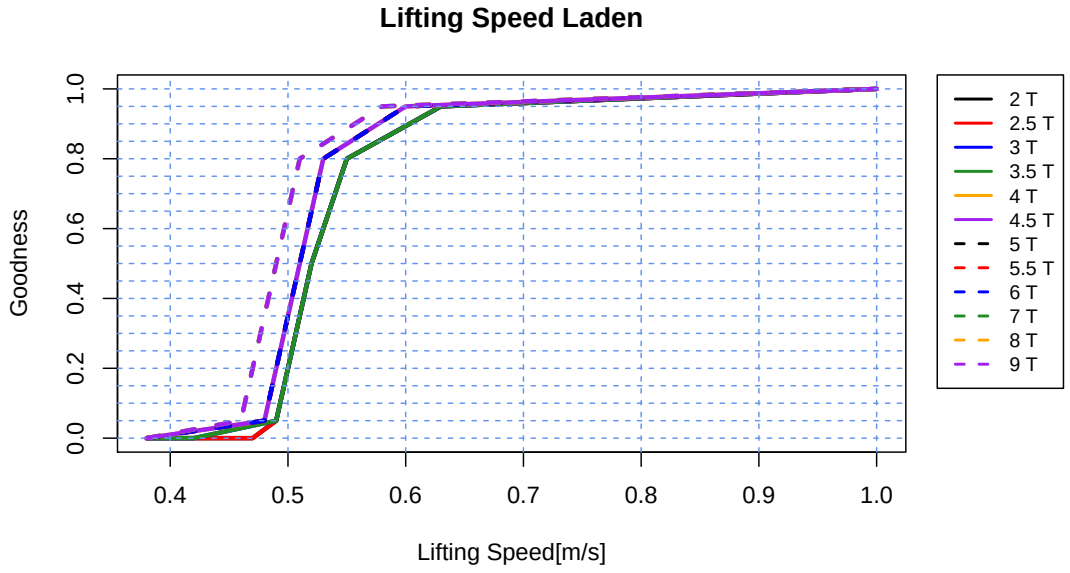


Figure 6.5: Lifting speed laden fuzzy membership set

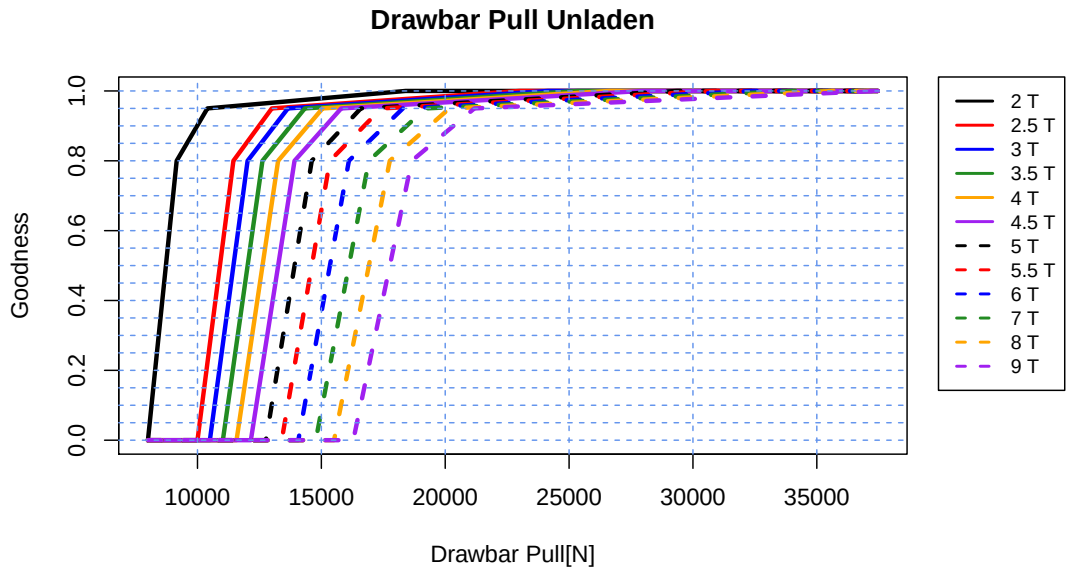


Figure 6.6: Drawbar pull unladen fuzzy membership set

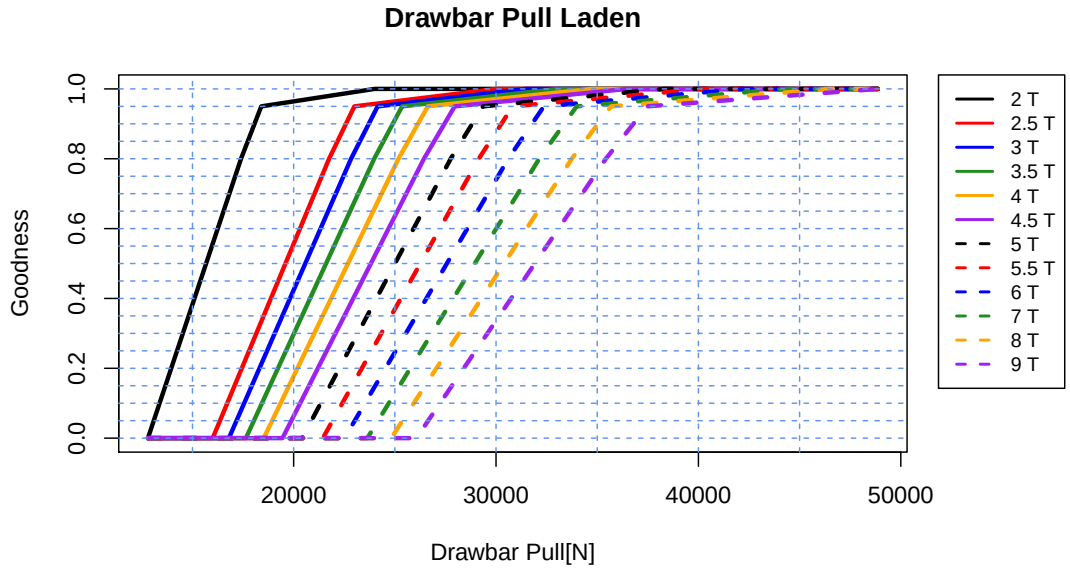


Figure 6.7: Drawbar pull laden fuzzy membership set

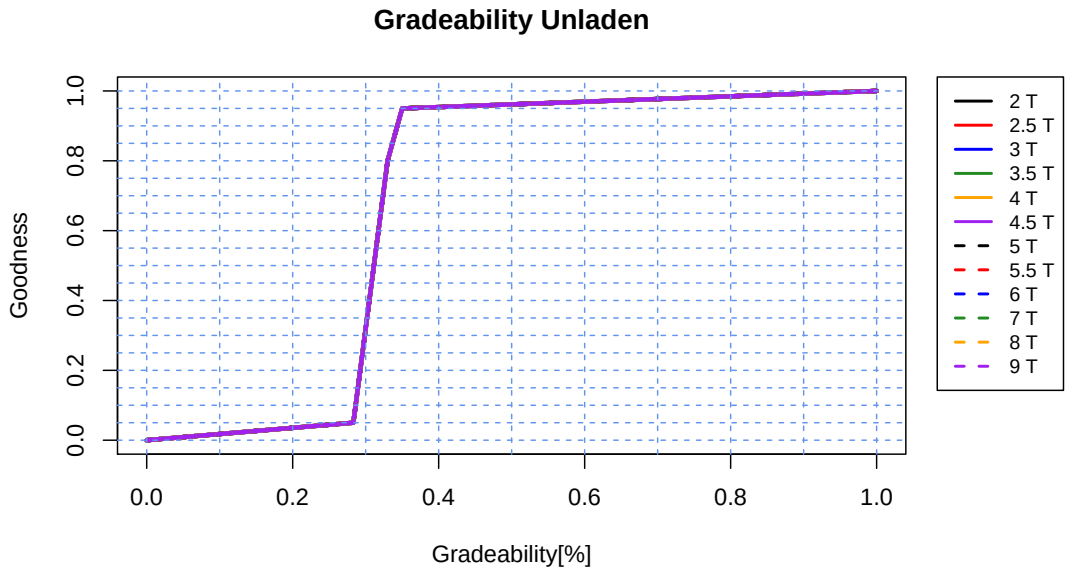


Figure 6.8: Gradeability unladen fuzzy membership set

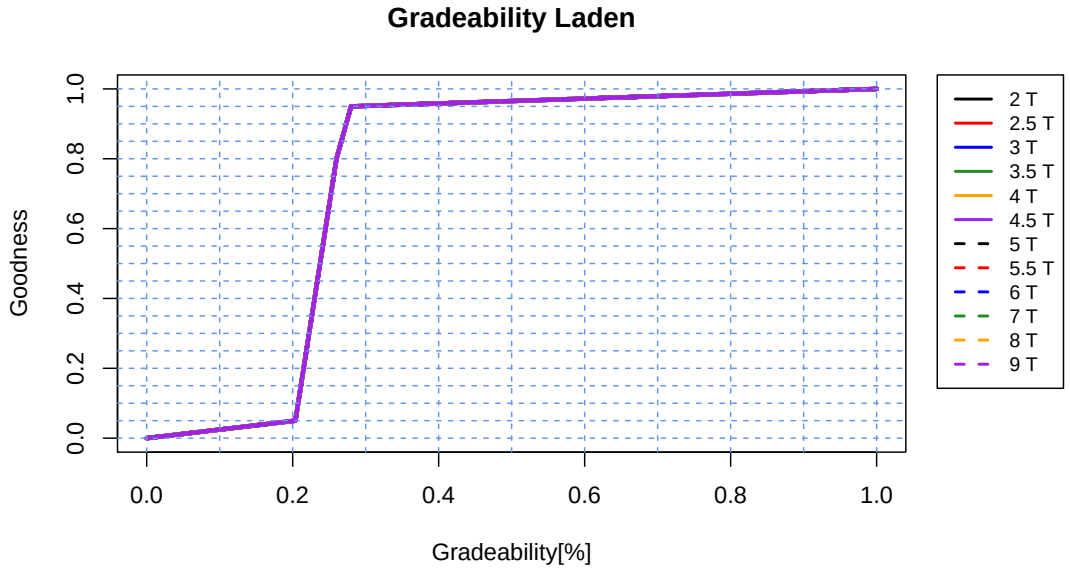


Figure 6.9: Gradeability laden fuzzy membership set

applications.

6.2.3 Aggregation of performance

As mentioned in the previous section, the original number of objectives is excessive and the solution for that is combining them in a certain way. For that, the entire performance of a product family was aggregated into a single objective. This objective will be part of the final problem, where other aspects apart from performance also need to be taken into account. The approach to aggregate the 108 performance attributes was to first aggregate the 9 attributes for each application to obtain a single value for each product, and then define an objective function for a candidate family composed by those products. This was shown graphically in chapter 5 in figure 5.6, and justified in the next subsection.

Product goodness

As indicated in chapter 5, the aggregation of performance for each product candidate was done with a formula that takes into account the relative importance of each attribute and the allowed degree of compensation between attributes (Dai and Scott, 2006).

$$G = \left[\frac{\sum_{i=1}^{9} w_i * Pg_i^s}{\sum_{i=1}^{9} w_i} \right]^{1/s} \tag{6.1}$$

The number obtained from that formula is multiplied by a function C to ensure that a variant with an unacceptable attribute is spotted and discarded:

$$C = \begin{cases} 0 & if P_i = 0, \ for\ any\ i \\ 1 & otherwise \end{cases} \tag{6.2}$$

The different weight parameters of each performance attribute for each applica-

tion and the compensation parameter were drawn from interviews with personnel from Sales & Marketing, exact details of those parameters are confidential and cannot be disclosed.

Justification for aggregation method

The problem of optimizing a product platform refers to balancing the advantages and disadvantages of component commonalization, and hence it is sensible to have two main objectives: what the product family will bring to us and what it will require from us. Or in a simple manner, how much income it will provide vs how much outgoing it will demand. The objectives will be defined in the section *Definition of objectives*. In this case study, a product family is composed of, a priori, 12 products, and the problem can be rephrased as *choose one of each product candidates to fill each of the 12 application placeholders*. Hence, it is advantageous to have a criterion to grade the performance of two different product candidates to be able to say which one is better in terms of performance, independently of the cost. This can be accomplished if a goodness attribute is defined for each product, comprising all the performance attributes.

6.3 Definition of the design pool

For this case study, the design pool is composed of 5 component types directly related to the selected performance features. They are the engine, transmission, tyres, hydraulic pump and lifting cylinder. the number of component instances is detailed in table 6.1.

Due to confidentiality issues, further details of the components cannot be disclosed in this thesis, but this will not affect the results.

Each possible combination of components for each application is a **product candidate**, and the set of all candidates is called **pool of product candidates**.

Engine	Six different engines named 1 to 6
Pump	Nine different pumps named 1 to 9
Transmission	7 transmission ratios from 14:1 to 20:1 in steps of 1:1
Cylinder	14 cylinder diameters between 65 mm and 130 mm in steps of 5 mm
Tyres	Five different tyres named 1 to 5

Table 6.1: Component options

When referring to the complete family, a **family candidate** is a combination of product candidates to form a product family, and the **pool of family candidates** is the set of all the possible family candidates. The pool of product candidates includes $12 * 6 * 9 * 7 * 14 * 5 = 317520$ candidates, or 26460 for each application. And the pool of family candidates $26460^{12} = 1.18 * 10^{53}$ candidates. This figures provide a clear idea of why assessing each possible candidate is not feasible.

6.4 Product simulation to find performance

This section describes the models used to calculate the different performance attributes of a product candidate as a function of the components used and the application for which it is intended.

6.4.1 List of symbols

The following is a list of symbols used in the description of the models.

f_i	Instantaneous fuel consumption in kg/s
ρ_{fuel}	Fuel density in kg/l
v	Driving speed in km/h
v_{max}	Maximum linear speed in km/h
rpm	Engine rotational speed in rpm
$Gear$	Overall gear reduction ratio

T_R	Tyre radius in m
τ	Engine torque in Nm
τ_{drive}	Engine torque required for driving
τ_{lift}	Engine torque required for lifting
μ_{roll}	Rolling resistance coefficient
η_{drive}	Efficiency of the driveline
M	Mass of the truck unladen in kg
g	gravity = $9.81 m/s^2$
P_{eng}	Engine power in W
P_{pump}	Hydraulic pump power in W
P_{drive}	Driving power demand in W
P_{lift}	Lifting power demand in W
η_{mech}	Mechanical efficiency of the pump
η_{vol}	Volumetric efficiency of the pump
η_{drive}	Drivetrain efficiency
η_{tc}	Torque converter efficiency
Q	Hydraulic flow in m^3/s
$Disp$	Pump displacement in cm^3/rev
p_{avail}	Maximum pressure available from the pump in Pa
Cyl_{area}	Cylinder cross sectional area in m^2
Cyl_{diam}	Cylinder diameter in m
p	Pressure in Pa
p_{loss}	Pressure losses in Pa
k_{hyd}	Constant for the pressure losses in kg/m^7
$F_{liftavail}$	Maximum available lifting force in N
DBP	Drawbar pull in N
$TC_{spratio}$	Speed ratio between the torque converter input and output

$TC_{tqratio}$	Torque ratio between the torque converter input and output
$Grad$	Gradeability in tangent of the maximum climbing angle

6.4.2 Fuel consumption

The fuel consumption considered is in accordance with the VDI (Verein Deutscher Ingenieure) 2198 cycle (VDI, 2012), which is a German standard and the de facto standard used for the specification sheets of all industrial trucks in Europe. The test track is shown in figure 6.10 and consists on two loading bays separated by 30 meters. The cycle is symmetrical and the truck carries its rated load throughout the cycle, i.e. there is no loading/unloading. The truck drives from the left end to the bay B. Once there the load is lifted to 2 meters, the mast tilts forward and backwards and then the load is lowered again. The truck reverses to the right end and drives to the bay A, where the same handling actions are performed, and the truck then reverses to the left end. This completes a cycle. The declared fuel consumption is the amount of fuel required to complete 60 cycles in 1 hour. The driver has to adapt the speed to complete an average of 1 cycle per minute, what for an experienced driver means driving slightly slower than what they can do.

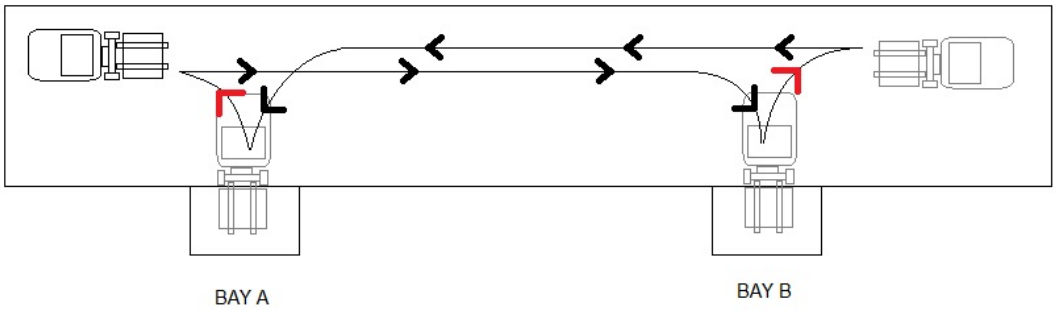


Figure 6.10: VDI2198 cycle

There are two main ways to simulate a driving cycle: forward and backward

modelling (Murtagh, 2015). The former consists on starting from the driver actuation and calculating through the powertrain to the tyres and handling apparatus, and the latter assumes a cycle and works back the demand on the powertrain and required fuel to accomplish it. The forward method is ideal to estimate performance and it is the method used for all the models in this case study except fuel consumption, whereas the backward method suits fuel calculation better, as it starts from a predetermined cycle. The model architecture for calculating the fuel consumption is shown in figure 6.11, and it is based on two power demand profiles, one for traction and one for lifting, and a speed profile. The profiles are based on test data obtained from a 2.5 tons truck and scaled up or down for different capacity trucks. The drive and speed profile go through a model of the tyres, gearbox and torque converter to calculate the torque and rpm demand on the engine. To that demand, the hydraulic demand is also added, calculated from a model of the mast with parametrized cylinders and pump. The engine model consists of two tables, a maximum torque-rpm and a fuel map depending on torque and speed. The maximum torque curve is also fed back to the demand profiles to check whether the engine is capable of following the cycle. If it is not, that means that the truck is either underpowered or the drivetrain, pump, etc. are not adequate. In this case the model issues a flag that will later be interpreted by the value model as a non-useful configuration. If the required torque is within the possibilities of the engine, then the torque and speed are fed into an engine specific fuel consumption map that calculates the instant fuel consumption. This value is integrated over a full VDI cycle and then multiplied by 60 to obtain the fuel consumed in an hour.

The relevant equations are:

$$VDIconsumption = \frac{1}{\rho_{fuel}} * 60 * \int_0^{60} dt * f_i(\tau, rpm) \tag{6.3}$$

This is 60 cycles, or one hour as specified by the standard VDI 2198. The integral

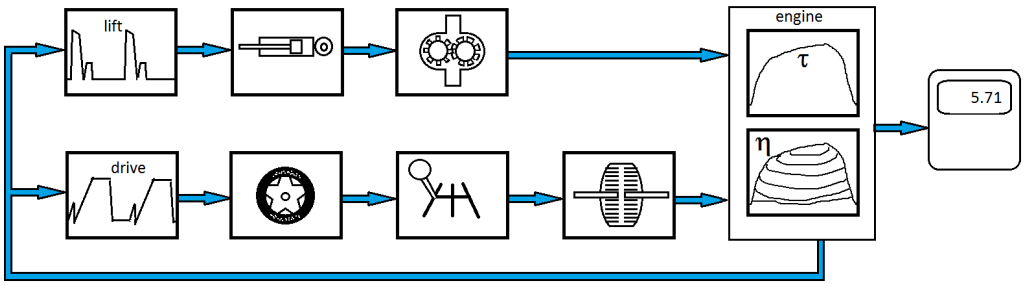


Figure 6.11: Fuel model diagram

over 60 seconds is the fuel spent for one cycle, and the function integrated is the instantaneous fuel consumption, which is obtained from a fuel map where the entry variables are the torque and engine speed. The whole result is then divided by the fuel density to convert from kilograms to litres, as this is the unit specified in the standard.

$$\tau = \tau_{pump} + \tau_{drive} \tag{6.4}$$

This is the total torque demanded from the engine at each instant. It is the sum of the torque required to drive and the torque required to lift, and the values are considered at the engine output, i.e. they account for the useful torque and also all the losses in the respective power transmission chains.

$$\tau_{pump} = \frac{P_{lift} * \frac{1}{\eta_{mast} * \eta_{mech} * \eta_{vol}}}{rpm * \frac{2*\pi}{60}} \tag{6.5}$$

where

$$\eta_{mast} = \eta_{mast}(Q, p), \quad \eta_{mech} = \eta_{mech}(Q, p), \quad \eta_{vol} = \eta_{vol}(Q, p) \tag{6.6}$$

The torque demand for lifting is the required theoretical power to lift the load divided by the different efficiencies of the mast mechanism and pump, and by the

engine speed. The efficiency of the mast mechanism is obtained empirically as a table where the entry variables are lifting speed and load weight. For the pump there are two different efficiencies: volumetric and mechanical. The former refers to the ratio of the oil that is pumped to the system respect to the incoming oil in the pump. Since the typical pump is a gear pump, there is some oil that is returned to admission pipe and the volumetric efficiency is never 100%. The latter refers to the mechanical losses due to friction of the different parts of the pump. Both efficiencies are provided by the pump manufacturers in tables with two entries, pressure and speed.

rpm when lifting is

$$rpm = \frac{Q * 6 * 10^7}{Disp} \ , \quad Q = \frac{v_{lift} * Cyl_{area}}{2 * \eta_{vol}} \tag{6.7}$$

The flow necessary to follow the cycle is the cylinder cross section multiplied by the lifting speed and divided by the pump volumetric efficiency and also by 2 due to the mast mechanical advantage, i.e. the load is lifted 2 meters for each meter of stroke coming from the cylinder. This can be seen in figure 6.12. Once the flow is known, it is divided by the pump displacement to calculate the required rotational speed. The extra numerical factors are placed in the equation only to match the units.

And rpm when driving is

$$rpm = \frac{v}{3.6 * T_R} * Gear * \frac{60}{2\pi} * \frac{1}{TC_{spratio}} \tag{6.8}$$

$$\tau_{drive} = \frac{P_{drive} * 60}{\eta_{drive} * \eta_{tc} * rpm * 2\pi} \tag{6.9}$$

The engine speed is the vehicle linear speed multiplied by the gear ratio and divided by the tyre circumference and the torque converter slip. The units are con-

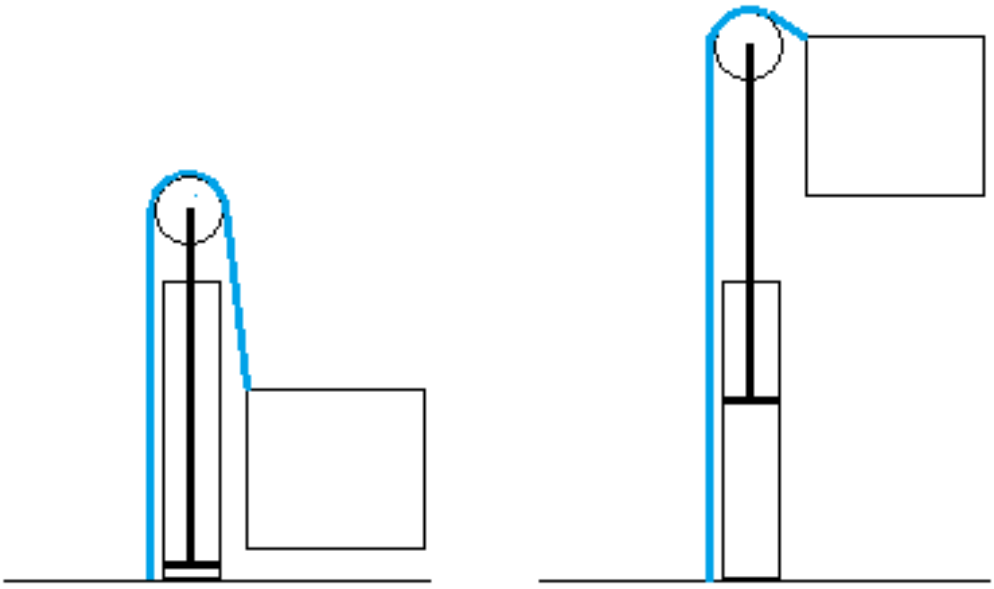


Figure 6.12: Typical mast mechanical advantage

verted to rpm for convenience. The torque required for traction is the instantaneous power read from the cycle profile divided by the engine speed and the power train and torque converter efficiency. The torque converter efficiency is a table depending on slip provided by the manufacturer.

6.4.3 Maximum speed

The maximum speed for a vehicle occurs at the point at which the resultant of the forces between the tyre and the ground is not capable of producing any acceleration, i.e. it is zero. The forces opposing the movement are the rolling resistance between tyres and ground, internal friction and aerodynamic drag. For road vehicles aerodynamic drag is the most important contributor to limiting the speed as the force grows with speed roughly linearly until approximately 50 mph and form then to the square of the speed. However, for the typical speeds of industrial trucks -around 10 to 15 mph- this force is very weak compared to the rolling resistance and can be neglected. The rolling resistance on the other hand, is nearly constant for any

speed, depending only on the tyre load, composition, pressure and state of wear, and the ground surface. Only the internal friction and rolling resistance need be considered as opposing forces for this type of vehicles. The model calculates the maximum driving speed laden and unladen based on the engine torque-rpm characteristic curve, the gearbox, the tyre radius and the rolling resistance. The engine is accelerated from 1000 rpm to its maximum speed with wide open throttle. The tractive force is calculated considering all the drivetrain reduction ratios and efficiencies and compared to the rolling resistance. The model calculates the speed based on the engine rpm, torque converter speed ratio, the gear reduction and the tyre radius and records the maximum value achieved before the tractive force becomes lesser that the rolling resistance, which is expected to happen at some point of the torque curve as it decays at the end of the range. The maximum speed is different for the laden and unladen cases. this is due to two reasons:

- The rolling resistance is proportional to the mass, within the tyre load range. So a laden truck will stop accelerating at a lower rpm than the same truck unladen.

- The tyres are not ideal incompressible circumferences, and the effective radius decreases for higher loads. This is itself due to two different effects:

  - Compression making the radius shorter

  - More tyre slip required for higher tractive force, resulting in a reduction of the linear speed for a given rotational speed.

Figure 6.13 shows the engine torque curve (black) and the required torque to overcome the rolling resistance laden (dashed green) and unladen (solid red). Points L and U show the rpm for those maximum speeds respectively. The ratio between the linear speeds is further decreased by the variation of the tyre radius.

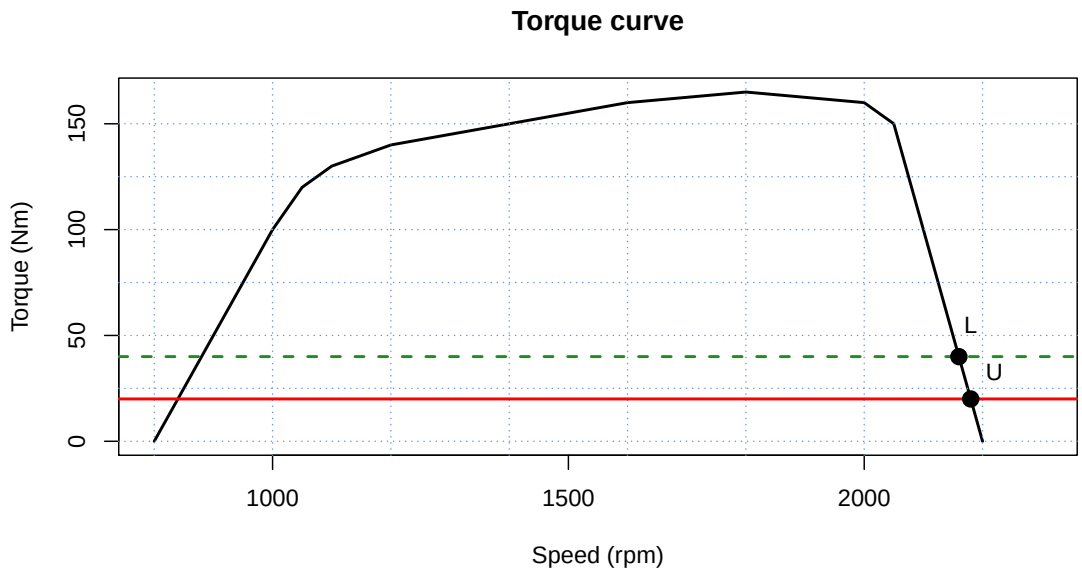


Figure 6.13: Engine rpm for which the laden and unladen maximum speeds are achieved

The relevant equations are:

$$v_{max} = 0.06 * max(\frac{rpm}{Gear} * 2 * \pi * T_R) \tag{6.10}$$

subject to the condition

$$\frac{\tau * Gear * \eta_{drive}}{T_R} > M * g * \mu_{roll} \tag{6.11}$$

where

$$\tau = \tau(rpm) \tag{6.12}$$

Equation 6.10 is a function that relates linearly the speed with the engine rpm. The *max* means that the value is the maximum that is acceptable under the condition stated in the equation 6.11. That equation requires the engine torque multiplied by the gear ratio and drivetrain efficiency and divided by the tyre radius to be bigger than the rolling resistance. The torque converter efficiency and slip is not considered for the calculation of the maximum speed because at that speed it is rigidly engaged,

i.e. there is neither slippage nor torque multiplication.

6.4.4 Lifting speed

The model that calculates the maximum lifting speed is based on the engine torque-rpm curve, the pump (displacement, volumetric and mechanical efficiencies), the lifting cylinder area, the hydraulic losses, the masses being lifted, the friction losses and the mast geometry. The engine is accelerated from 1000 rpm and both the power and rpm are fed to the hydraulic pump. The pump calculates the flow and the maximum pressure that can supply at that flow. For this two lookup tables are required, volumetric and mechanical efficiency depending on pressure and rpm. The rpm is directly fed from the engine, but the pressure comes from the end of the model through a 0.01 seconds transport delay, to prevent algebraic loops. The hydraulic losses depending on the flow are subtracted from the maximum pressure available, and the resulting pressure is multiplied by the cylinder cross sectional area to calculate the maximum lifting force available. This lifting force is compared to the force required to lift the load and overcome the friction of the mast, and when it becomes greater, the model stops.

To calculate the lifting speed, the flow is divided by the cylinder cross sectional area and multiplied by two to reflect the pulley-chain factor. The minimum of the available lifting force and the required force is divided by the cylinder cross sectional area to calculate the real pressure. Then it is added to the pressure losses to calculate the real pump pressure, which is fed to the pump efficiency lookup tables as previously indicated. The relevant equations are:

$$P_{eng} = \tau * rpm * \frac{2\pi}{60} \tag{6.13}$$

$$P_{pump} = P_{eng} * \eta_{mech} * \eta_{vol} \tag{6.14}$$

The power delivered by the engine is the torque multiplied by the speed. That power is multiplied by the pump mechanical and volumetric efficiencies to obtain the effective hydraulic power.

$$Q = Disp * rpm * \eta_{vol} * \frac{1}{6 * 10^7} \tag{6.15}$$

The flow is the pump displacement multiplied by the engine speed and the pump volumetric efficiency. The engine speed and the pump speed is the same as they are rigidly engaged without any gear reduction.

$$p_{avail} = \frac{P_{pump}}{Q} \tag{6.16}$$

The pressure is the effective power divided by the flow. The cylinder area is just a cross section calculation with the diameter, which is the main specification for each cylinder.

$$Cyl_{area} = (\frac{Cyl_{diam}}{2})^2 * \pi \tag{6.17}$$

$$p_{loss} = k_{hyd} * Q^2 \tag{6.18}$$

Pressure losses are proportional to the square of the flow. This is an empirical relation, and k_{hyd} is also an empirical constant.

$$F_{liftavail} = (p_{avail} - p_{loss}) * Cyl_{area} \tag{6.19}$$

Finally the force is pressure differential multiplied by the cylinder cross section.

6.4.5 Drawbar pull

The model that calculates the drawbar pull is based on the engine torque-rpm curve, the torque converter characteristics, the tyre radius and the drive axle load. The engine is accelerated from 1000 rpm to maximum speed and the rpm is compared to the rpm at the torque converter output shaft corresponding to 1.6 km/h at the drive tyres to find the speed ratio between the input and output of the torque converter. A look up table maps the speed ratio into a torque ratio, which is multiplied by the driveline efficiency and the gear ratio to find the maximum torque at the wheels. That maximum torque is divided by the tyre radius to yield the maximum tractive force. The maximum value during the run is stored. The maximum tractive force allowed for the tyres is calculated as the axle load multiplied by the tyres friction coefficient. The minimum of this and the tractive force previously calculated is the drawbar pull. The relevant equations are:

$$TC_{spratio} = \frac{Gear * 60 * 0.444}{rpm * 2\pi * T_R} \tag{6.20}$$

The torque converter slip ratio is the relation between the output speed, i.e. the vehicle speed - 1.6 km/h - multiplied by the gearbox ratio and divided by the tyre circumference, and the input speed, or engine speed. A table provided by the manufacturer converts the slip ratio to the torque ratio.

$$TC_{tqratio} = f(TC_{spratio}) \tag{6.21}$$

$$DBP = \frac{\tau * TC_{tqratio} * Gear * \eta_{drive}}{T_R} \tag{6.22}$$

The traction torque is the engine torque multiplied by the torque converter torque ratio and the drivetrain efficiency. This torque divided by the tyre radius is the tractive force, or drawbar pull.

0.444 is the required speed of 1.6 km/h given in m/s. The function that relates the torque ratio with the speed ratio is characteristic of each torque converter.

6.4.6 Gradeability

The model that calculates the gradeability is based on the output from the drawbar pull after adding a new function for the axle loads, the mass of the truck and the rolling resistance. The steepest slope that a vehicle can climb is that for which the tractive force equals the opposing force. The tractive force is calculated by the drawbar pull model, taking into account the variation on the drive axle load due to the angle.

The rolling resistance is subtracted from the drawbar pull and the result divided by the weight (force) of the truck. The arcsine of that is the maximum angle that can theoretically be surmounted, and the gradeability declared as the tangent of that angle. The relevant equations are:

$$Grad = \tan\left[\sin^{-1}\left(\frac{DBP - \mu_{roll} * M * g}{M * g}\right)\right] \tag{6.23}$$

where

$$DBP = DBP(angle) \tag{6.24}$$

Since the main equation is implicit, the angle solution is required to calculate the DBP, the solution is found by inputting progressively bigger angles to the right hand of the equation until the left hand matches the input angle. This is equivalent to simulating the vehicle driving over an increasing slope until it stops.

6.4.7 Solver

For this case, the selected solver was ode3 (Bogacki-Shampine). This is an ODE, fixed step, continuous variables, explicit, non-stiff, third order solver that uses the

Bogacki-Shampine method (Mathworks, 2018). The choice of solver is due to the author's preference after the experience with this particular solver for similar problems. The use of another solver would not have brought any change in the results, as long as they feature continuous variables and are explicit, which are two main characteristics of the models as they were written.

6.5 Definition of objectives

The definition of the problem objective requires a conversion from performance figures for all the products in the family to how good or desirable that product family is. There are several ways to define those objectives, and identifying the most suitable one requires knowledge of the specific problem and judgement.

As previously stated, for this case study two objectives have been selected.

- Objective 1: A measure of what the product family will provide - Revenue.
- Objective 2: A measure of what the product family will cost - Cost.

A potential third objective can be the time and effort to develop the family. Although this can also be absorbed by the second objective.

6.5.1 Objective 1 - Revenue

This objective considers the gross revenue that can be expected from the product family depending on how good the family is. Gross revenue is the sum of the prices for which all the products are sold over a time period. For this case study, the time period considered is one year. There is a complex relation between price, number of products sold, and a third variable that is the product appeal. For this case study, the independent variable is the product appeal, which is the variable that depends directly on the product performance. And the objective is the product of prices and

number of sales. This product is also a function that can be maximized, but for this case study it will be considered that the market share is a hard target, i.e. it is required to sell a particular number of units of each product, while the price of each unit will depend on the appeal, i.e. the independent variable. This is written in the equation

$$\begin{aligned} Revenue &= \sum_{i=1}^{n} Pr_i * Np_i \\ Np_i &= constant, \quad Pr_i = f(G_i) \end{aligned} \tag{6.25}$$

where Pr_i is the price of each product, Np_i is the number of units to be sold for each product, and $f(G)$ is a function of the goodness of each product

For the case study, the amount of trucks to be sold from each capacity were entered as a 12 element vector Np. And another 12 element vector $Pmax$ with the maximum prices for what an ideal truck for each capacity (individual performance value = 1) could hypothetically be sold.

For each family candidate selected, there is a vector named $Gfam$ with the individual goodness of each of the trucks that compose the family. Then this vector is passed to the function $f(G)$ to obtain a vector of prices Pr for what the products of that given family candidate can be expected to sell. The element by element product of Pr and Np is the vector of gross revenue for each product, and the sum of this vector elements is total gross revenue, or objective 1.

As a clarification, this gross revenue is valid only for the purposes of optimizing the component commonality across the product family, and not to be confused with a realistic revenue for economical forecasting purposes.

The function $f(G)$ used in this case study is simply a linear relation between the goodness and the price. The development of a more accurate or realistic function would require a joint sales and marketing effort and it is a long problem on itself of

a more commercial nature, out of the scope of this thesis.

6.5.2 Objective 2 - Cost

This objective considers the costs associated with each family candidate that depend on the choice of components for each product. Any other cost is omitted, and the objective cannot be considered as a true cost for economic purposes.

For this case study, the costs considered are those of the parts, and take into account the prices and also how the prices change with increasing volumes.

For each component, a curve was defined to relate the number of them used in the whole family with the price per unit. The curves were modelled following a law of exponential decay.

$$P_{unit} = P_0 + P_0 * e^{-k(u-1)} \tag{6.26}$$

Where P_{unit} is the price per unit, P_0 the minimum price per unit for an infinite number of units, u the number of units and k a constant to define how fast the curve approaches the minimum value. Large values of k correspond to curves that get close to the minimum for a relatively low number of components, whereas small values of k correspond with components that require a large volume in order to have noticeable effects in the price per unit. The -1 in the exponential is added to make it easier to define the price for a one-off unit, which in the equation as it is defined is double the limit for an infinite amount of units. This can be easily changed by adding a multiplicative factor to the exponential. As an example, figure 6.14 shows the price curve of one particular engine.

The objective 2 is the sum of the purchasing prices of all the components necessary to build the candidate family.

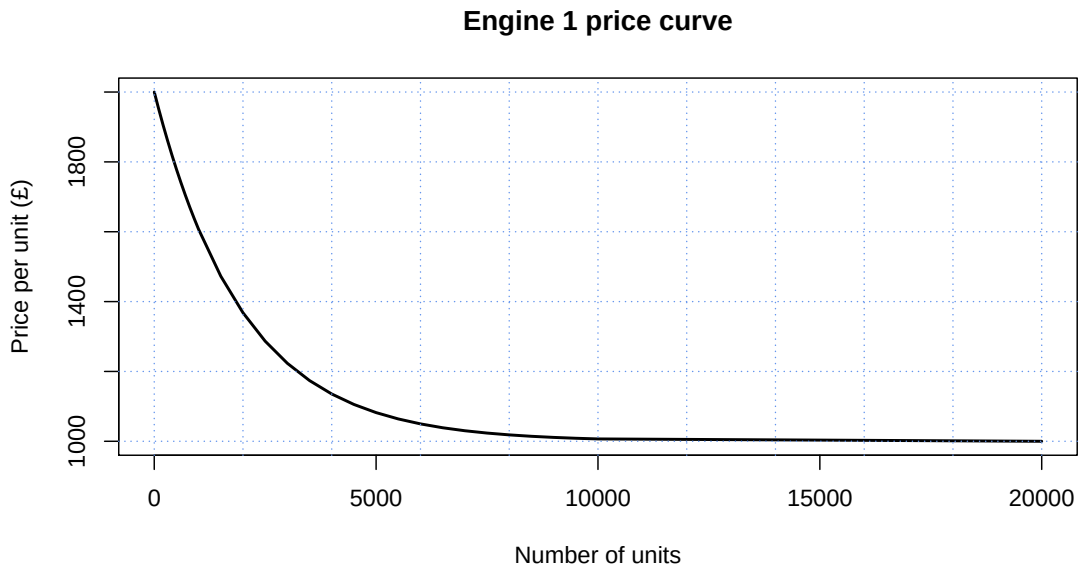


Figure 6.14: Example of price vs number of units

$$Cost = \sum_{i=1}^{5} \sum_{j=1}^{ct} P_{unit,i,j} * N_{units,i,j} \qquad (6.27)$$
$$P_{unit,i,j} = f(N_{units,i,j})$$

Where i is the component type, j the component instance, ct the number of different component instances for each component type, $P_{unit,i,j}$ the price to purchase each component instance j of the component type i, and $N_{units,i,j}$ the number of component instance j of the component type i. $P_{unit,i,j}$ is a function of $N_{units,i,j}$.

The model has a matrix B, with the minimum prices P_0 for each component instance and another matrix K, with the coefficients k for each component instance. The first column refers to the engines, the second column to the pumps, etc. For this case study, the matrices are 14x5, 5 columns one for each component type and 14 rows for the maximum number of instances for a component type, in this case cylinders. The matrix is filled with zeros where necessary, as there are not the same number of options for each component.

6.6 Optimization

This is the process of finding the best, or a good product family according to the defined objectives.

The problem in this case study can be formally stated as

$$\begin{aligned} \underset{x}{Minimize} \quad & \mathbf{F}(\mathbf{x}) = [-F_1(\mathbf{x}), F_2(\mathbf{x})]^T \\ & F_1, F_2 : \mathbb{R}^{12x5} \rightarrow \mathbb{R}, \forall \mathbf{x} \in S \subseteq \mathbb{R}^{12x5} \\ F_1 = \sum_{i=1}^{12} Pr_i * Np_i, \quad & F_2 = \sum_{i=1}^{5} \sum_{j=1}^{ct} P_{unit,i,j} * N_{units,i,j} \\ & Pr_i = f(G_i) \\ & subject\ to\ G_i \neq 0 \quad \forall i \end{aligned} \tag{6.28}$$

Where F_1 and F_2 are the two objectives and F_1 has a negative sign for consistency in the equation, since the original function has to be maximized. $\mathbf{x}$ is any solution represented as a 12x5 element matrix where the rows are applications and the columns components. $\mathbf{x}$ is a member of the design space S defined as

$$\begin{aligned} S_{i,j} \in \mathbb{N}, \quad \forall i \in [1, 12] \\ S_{i,1} \leq 6 \\ S_{i,2} \leq 9 \\ S_{i,3} \leq 7 \\ S_{i,4} \leq 14 \\ S_{i,5} \leq 5 \end{aligned} \tag{6.29}$$

6.6.1 Development of optimization process

The algorithm that will be used is a genetic algorithm. They are well suited for combinatoric problems of this type (Simpson and de Souza, 2004). The details of the algorithm as well as the results will be presented in the next chapter.

The general workflow diagram for a genetic algorithm is shown in the figure 6.15. The steps are:

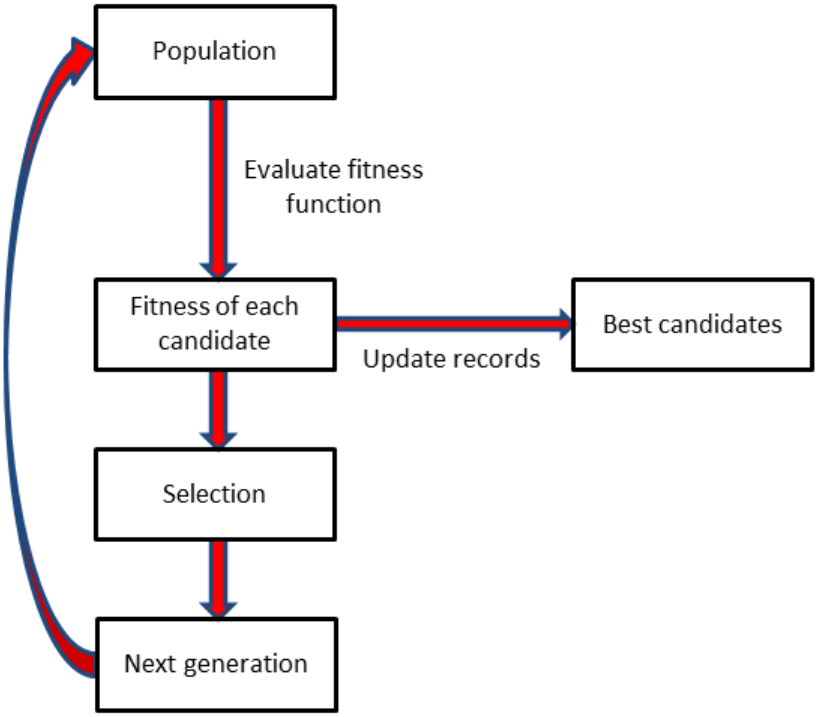


Figure 6.15: Genetic algorithm workflow

- *Step 1*: Have a population of candidate solutions
- *Step 2*: Evaluate the fitness function for all the candidate solutions in the population
- *Step 3*: Select the best candidates in the population according to the algorithm

parameters, and update the record of the best solutions. This record can be a single candidate for single objective algorithms or a Pareto set for multi-objective algorithms

- *Step 4*: Obtain a new generation of candidates by combining selected candidates, mutations to existing candidates and introduction of new random candidates

This process involves evaluating the fitness function for each candidate solution in the generation at every iteration of the algorithm. For the case study this would involve calculating the 9 performance attributes of each of the 12 products of the candidate family. This calculation is the most time consuming part of the iteration. To illustrate this point, with the particular laptop used for this research, it took around 42 seconds to simulate the performance of a family candidate, whereas the remaining steps: fuzzification and calculation of objectives took a fraction of a second. At this pace, the algorithm would be able to evaluate approximately 85 family candidates per hour. Fortunately, there are ways to improve the efficiency of the searching algorithm, such as downsizing the problem and varying the work flow.

Downsizing the problem

With 6 engines, 9 pumps, 7 transmission ratios, 5 tyres and 14 cylinders, the number of possible combinations for a single truck is

$$6 * 9 * 7 * 5 * 14 = 26460 \; combinations \; for \; each \; individual \; variant \tag{6.30}$$

And since there are 12 variants, the total number of combinations to construct a product range is

$$26460^{12} = 1.178 * 10^{53} \; possible \; ranges \tag{6.31}$$

This number of combinations includes some impossible families, such as for example one in which the 9 ton truck has a cylinder smaller than the 2 ton truck. These kind of restrictions are easy to identify and can drastically reduce the search space if they are removed beforehand. Just by eliminating one candidate from one application, the searching space is reduced by $4.451 * 10^{48}$ family candidates. It is easy to see in an intuitive way the benefit of spending some time in eliminating obviously impossible product candidates, while at the same time it is important to eliminate only those combinations that are absolutely sure to be impossible, to avoid eliminating potentially good solutions. An exercise to reduce the pool was done and achieved the final pool of candidates detailed in table 6.3

2.0 tons	270 combinations
2.5 tons	540 combinations
3.0 tons	900 combinations
3.5 tons	810 combinations
4.0 tons	1296 combinations
4.5 tons	1296 combinations
5.0 tons	1512 combinations
5.5 tons	1296 combinations
6.0 tons	1080 combinations
7.0 tons	1080 combinations
8.0 tons	864 combinations
9.0 tons	486 combinations

Table 6.3: Number of possible truck designs for each capacity after eliminating all the options identified as non feasible by engineering judgement

Then the total number of possible ranges is reduced to $1.713 * 10^{35}$, which represents an improvement of a factor of $6.87 * 10^{17}$. An impressive reduction but still far from making the problem solvable by brute force.

Varying the workflow

Each possible product is represented as a 5-element vector, for example (1,2,4,2,3) refers to a product designed with engine (e) #1, pump (p) #2, gear (g) #4, tyres (t) #2 and cylinder (c) #3. Each family candidate is then represented as a 12x5

matrix in which each row is the design for each capacity from 2 tons to 9 tons in the pre-determined steps. This is illustrated in the matrix below.

$$\begin{bmatrix} e_{2.0} & p_{2.0} & g_{2.0} & t_{2.0} & c_{2.0} \\ e_{2.5} & p_{2.5} & g_{2.5} & t_{2.5} & c_{2.5} \\ e_{3.0} & p_{3.0} & g_{3.0} & t_{3.0} & c_{3.0} \\ e_{3.5} & p_{3.5} & g_{3.5} & t_{3.5} & c_{3.5} \\ e_{4.0} & p_{4.0} & g_{4.0} & t_{4.0} & c_{4.0} \\ e_{4.5} & p_{4.5} & g_{4.5} & t_{4.5} & c_{4.5} \\ e_{5.0} & p_{5.0} & g_{5.0} & t_{5.0} & c_{5.0} \\ e_{5.5} & p_{5.5} & g_{5.5} & t_{5.5} & c_{5.5} \\ e_{6.0} & p_{6.0} & g_{6.0} & t_{6.0} & c_{6.0} \\ e_{7.0} & p_{7.0} & g_{7.0} & t_{7.0} & c_{7.0} \\ e_{8.0} & p_{8.0} & g_{8.0} & t_{8.0} & c_{8.0} \\ e_{9.0} & p_{9.0} & g_{9.0} & t_{9.0} & c_{9.0} \end{bmatrix}$$

In a general genetic algorithm such as that one described in figure 6.15, the fitness function is evaluated for all the members of the population at each iteration. In this case, since each product candidate can be part of many families and then it is likely to appear several times during the search, that would involve repeating the same simulation many times. And since it has been observed that the simulation is the longest part of the iteration in terms of computational time, it can be advantageous to avoid that type of repetitions.

The method to avoid that repetition was to simulate all the possible products beforehand. The total number of possible products according to table 6.3 is 11430, and simulating the performance of each of them took around 9 hours with a Matlab script that passed the parameters of each product to the relevant Simulink models. This simulation exercise allowed a table to be written with the performance figures

for each attribute for each possible product candidate for each application. Those figures were also fuzzified and passed through the function to obtain the overall product goodness. The last two steps are insignificant in terms of computational time. Figures 6.16 and 6.17 show the difference between the original flow and the improved flow, where the searching algorithm iterates over the results of the process within the dashed line.

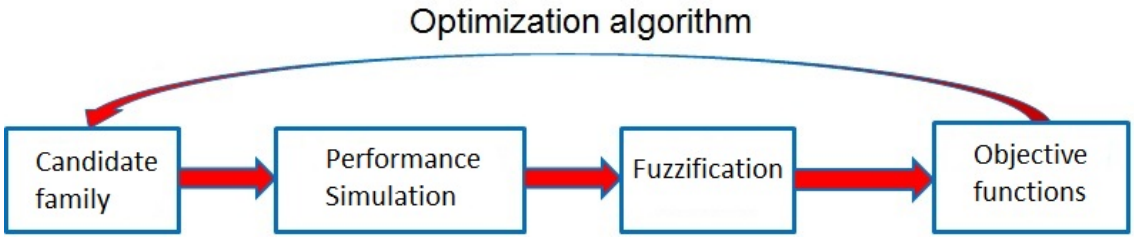


Figure 6.16: Original work flow

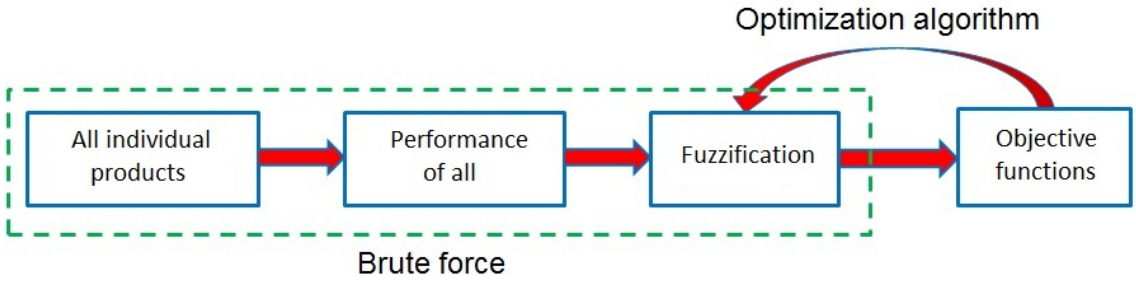


Figure 6.17: Alternative more efficient work flow

Some of the variants returned an individual goodness of zero due to the restriction of equation 6.2, and they were eliminated from the database of possible variants. The list was hence further reduced as shown in table 6.4. This further reduction leaves the total number of candidate families as $6.0 * 10^{29}$, which is a significant improvement.

Once the overall goodness of each product candidate are tabulated, the objectives function can evaluate in excess of 4000 family candidates per second, or $1.4 * 10^{7}$ per hour, which is 168000 times more than the original algorithm work flow. After spending the 9 hours necessary to tabulate those goodnesses, the new work flow

quickly reaches and leaves behind the performance of he original one. After 50 hours, the original work flow will have evaluated approximately 4300 family candidates out of $1.7 * 10^{35}$, i.e. 1 in every $4 * 10^{31}$. Whereas the new work flow will have done so to $7.2 * 10^{8}$ out of $6 * 10^{29}$, i.e. 1 in $8.3 * 10^{20}$. This represents a $4.8 * 10^{10}$ fold improvement. For longer times, the improvement factor increases as the initial 9 hours become relatively less important. The factor by which the new method is better than the original tends asymptotically to $4.8 * 10^{10}$, as shown by the parallel logarithmic curves in the figure 6.18

2.0 tons	221 combinations
2.5 tons	168 combinations
3.0 tons	265 combinations
3.5 tons	645 combinations
4.0 tons	629 combinations
4.5 tons	464 combinations
5.0 tons	561 combinations
5.5 tons	369 combinations
6.0 tons	258 combinations
7.0 tons	180 combinations
8.0 tons	216 combinations
9.0 tons	156 combinations

Table 6.4: Number of possible trucks designs for each capacity after discarding all those hat returned an individual performance value of zero

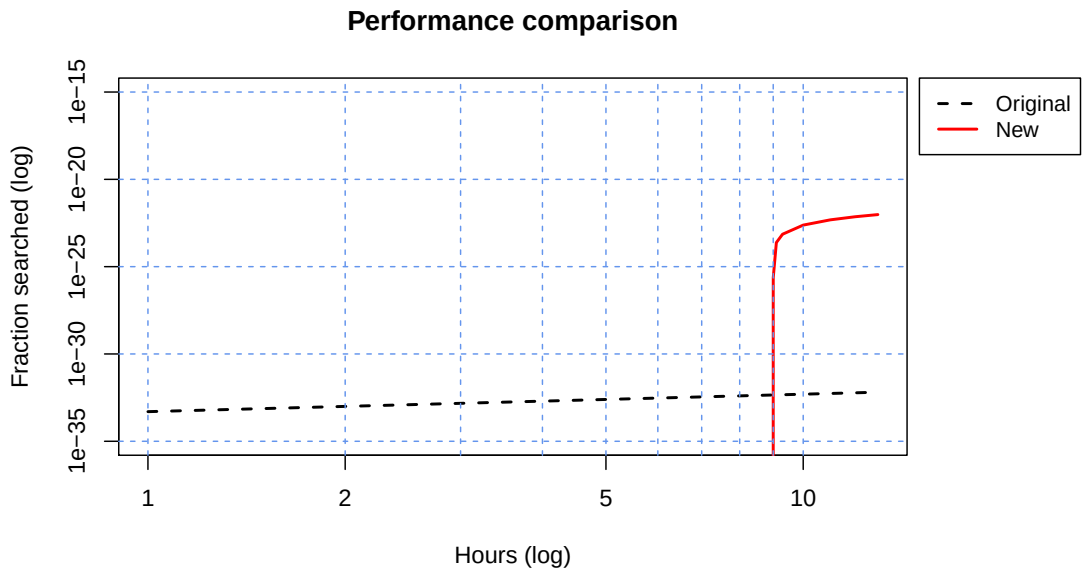


Figure 6.18: Performance comparison between the original and new method. Both axes are logarithmic scales

6.7 Summary

This chapter has shown the steps described in the previous chapter applied to a real case. It has also discussed the process by which the search space has been reduced and the flow improved before running the algorithm to improve the efficiency.

The next chapter will describe the low level details of the algorithm in terms of how the population is chosen and sized, how it is recombined after each generation and how the best candidates are stored. The results will also be shown.

Chapter 7

Implementation and results

Chapters 6 described the case study, the modelling, the definition of the objectives and introduced the searching algorithm. Now this chapter will first describe the searching algorithm in detail as it was implemented, and then show the results obtained. This chapter is one of a descriptive nature, the results will be shown but neither validated nor discussed. The next two chapters afterwards will respectively explain how the models were validated and provide a full discussion on the results and the implication for the general purpose of this thesis.

7.1 Algorithm in R. Detailed description

The searching algorithm is separated into three main parts:

- Generating the workspace
- Setting the parameters
- Running the algorithm

7.1.1 Generating the workspace

The first step is to load the data previously generated with the Simulink models. There are 12 csv files, one for each application, with all the possible combinations of components acceptable. Each row is one combination, and the columns are: engine, pump, tyre, gear, cylinder, fuel consumption, maximum speed unladen, maximum speed laden, maximum lifting speed unladen, maximum lifting speed laden, drawbar pull unladen, drawbar pull laden, maximum gradient unladen and maximum gradient laden. These databases are stored as *p20*, *p25*, *p30* , *p35* , *p40* , *p45* , *p50* , *p55* , *p60* , *p70* , *p80* and *p90*. These files have the performance data given in physical units.

The performance data in these p databases is given in physical units, and it has to be transformed into goodness. *Fvalues* is a vector with the fuzzy level cuts $(0, 0.05, 0.2, 0.5, 0.8, 0.95, 1)$, and *Fsetsdata* a matrix read from a csv file with the performance values corresponding to each level cut. *Fsetsdata* is divided in 12 blocks (as many as applications) with nine rows each, one row per performance attribute. The columns are the level cuts from 0 to 1, and each block is separated from the next one by a NA row.

The information in *Fsetsdata* is rearranged into a 9x7x12 array named *Fsets*. The indexes are:

- 1st index: Performance attribute
- 2nd index: Fuzzy level cut
- 3rd index: Application

e.g. The element $Fsets(3, 2, 5) = 17.0875$, this means that the maximum speed unladen to score 0.05 in a 4 tons application is 17.0875 km/h.

A matrix is created for each application to store how each possible product scores in each performance attribute. Those matrices are named *g20*, *g25*, *g30*, etc.

This is done by interpolating the performance in each attribute between the closest values for that application and that attribute in *Fsets* The original matrices with the performance values in physical units is no longer necessary and therefore removed from the workspace at this point.

Then a loop goes from bottom to top of each goodness matrix and deletes every row in which at least one attribute is zero, as that product is deemed unacceptable.

The algorithm now calls the function *Overall_goodness* for each remaining possible product of each application. The returned value is stored in 12 6-column matrices named *a20*, *a25*, etc. where the first 5 columns are the components that define a possible product and the 6th column is the overall goodness of that product. These are the data matrices required to run the searching algorithm, and together with *aNames* are kept in the workspace. All other variables can be cleared at this point to keep it neater.

7.1.2 Setting the parameters

The searching algorithm is a multi-objective genetic algorithm based on two different objectives. Each individual in a generation is a candidate family, i.e. 12 different products destined to fulfil one application each. There is no a priori articulation of preference and the outcome of the algorithm is a Pareto front. The first step is to define some parameters:

- sPopulation: number of individuals, or candidate families, in each generation
- uParents: number of individuals in each generation generated by combinations from existing parents
- Mutation: number of individuals in each generation coming from mutated parents. A mutated parent is a family candidate in which some of the components of some of the trucks are changed

- Mutation chance: chance of mutation for each gen of the parents to be mutated
- nIterations: number of generations over which the algorithm will be run
- nRecord: number of historical Pareto fronts to be stored

A 12 elements vector - *Sizes* - is created with the length of each application product database. This is necessary for the random selections that will occur later on.

Each individual is a 12 element vector in which each element is the number that a product occupies in its application database, e.g. the individual [35, 45, 65, 43, 45, 28, 76, 43, 46, 57, 86, 43] represents a candidate family composed of the product in row 35 for the first application, product in row 45 for the second application, etc. A generation is initially represented as a 12-column matrix in which each row is an individual. There are as many rows as the parameter *sPopulation* indicates.

The Pareto front is a 14-column matrix where the 12 first columns represent an individual with the convention explained above, and the latter two columns the score in the two objectives. Each row is one of the Pareto efficient solutions found. The Pareto front is initialized with one individual but can grow and shrink during the execution of the script, the size will be stored in the variable *sPareto*. The initialization is done with a random individual on which the function *Objectives* is applied. Generation of a random individual is done by creating a 12 elements vector in which each element is the ceiling of a random number between 0 and 1 multiplied by the size of the corresponding product database.

The first generation is created by generating a sPopulation number of random individuals. Then two columns with zeros are added to the end of the matrix in which the scores in the two objectives will be stored.

The final step before starting running the genetic algorithm is to define the *Pareto_historical* set and which iterations will be stored in it. Due to the nature

of the evolution of the found solutions, improving quickly at the beginning and slowing down later on, it makes sense to record the Pareto fronts at longer intervals as the algorithm advances in time. The chosen method was to record the front in a logarithmic scale. A vector named Milestones is defined with the iteration at which the fronts will be recorded. The Pareto historical is created as a nRecord x 2 matrix. Since the purpose of this record is only to show the evolution of the Pareto front, it will store only the scores in the objectives, and not the family candidates that achieved them. This matrix will grow in columns every time a recorded front includes more points than the current size. The variable *fill* is the next row of the Pareto historical that will be filled when the algorithm reaches the following milestone.

7.1.3 Running the searching algorithm

The main body of the algorithm is a loop that runs from 1 to the number of iterations previously selected. The first step is to run the function *Objectives* with the initial generation as input. The score in the two objectives is stored in columns 13 and 14 of the matrix *Population*. Then the individuals are sequentially arranged into four sequentially non-dominated fronts and the remainder of points. The first non-dominated front is the Pareto front obtained from that generation. The second non-dominated front is the Pareto front once the individuals belonging in the first non-dominant front have been removed, and so on. For the first non-dominated front two matrices are created named *Non_dominated1* and *Remainder*. A loop runs through each individual with a nested loop comparing its objectives with those of every other individual. The nested loop breaks if any point is found to be better at the two objectives than the point under test, meaning that that point is dominated. The point is then stored in *Remainder*. If the nested loop reaches the end without finding a point that dominates the point under test, then that is a non-dominated

point and it is stored in *Non_dominated1*. The points in *Remainder* then go through a similar process to define the second set of non-dominant points, and twice more until all the points are classified into 1st, 2nd, 3rd, 4th non-dominant fronts and a 5th set with the remaining points. The points belonging in the remainder after this process will be discarded for composing the next generation.

All the non-dominated fronts have 15 columns: 12 for the products that compose the candidate family, 2 for the objectives and a 15th column to give a weight depending on which non-dominated front they are. Those weights are 1, 0.8, 0.6 and 0.4 for the 1st, 2nd, 3rd and 4th non-dominated sets respectively. The weights are later on used to bias the probability of being reproduced in the next generation toward the best performers.

After this arranging process, the Pareto front is updated if any of the points in the current generation is not dominated by any of the previous Pareto points. This is done by checking the points of *Non_dominated1*, since only points in that set can be Pareto efficient. A loop goes through the points in *Non_dominated1* and a nested loop tests each of those points against every point in *Pareto*. The result of each match can be:

- Dominated: The point is not Pareto efficient and it is not stored in the Pareto matrix. The nested loop breaks
- Dominates: The point dominates the current point of the Pareto front, so this last point is removed from the set
- Redundant: The point is already repeated in the Pareto matrix, the loop breaks
- None of the above

If the outer loop reaches the end and the point is not dominated by any, then that point is added to the Pareto front matrix.

Once the Pareto front is updated, if the iteration number is one of those identified in *Milestones*, then the current Pareto set is also added to the Pareto historical set. the Pareto historical rows represent the Pareto fronts at the different time milestones. Each row has twice as many filled positions as the size of the Pareto front for that iteration, the odd positions correspond to the first objective and the even positions to the second objective.

Creating the next generation

Once the Pareto set has been updated, the next and last step of the iteration is to create the next generation. For that, the points in the four non-dominated fronts of the current generation are put together in a matrix named *Parents*. Then the 15th element of each row is multiplied by a random number between 0 and 1, and the rows re-ordered according to the resulting number. This way the selection is biased towards the points in the highest non-dominated fronts but all the points in 'Parents' have a chance to be selected.

The size of the Parents matrix can be greater or lesser than the parameter *uParents*, which is the preference for the number of children in a generation coming directly from previous individuals without mutation. The algorithm then generates the lesser of the two numbers as new individuals directly from existing parents. The method to create these individuals is a loop that chooses two parents from the Parents matrix, the first one sequentially and the second one randomly. Then the new candidate family is built element by element with a 50 % chance of being selected from either parent.

The next group of children to be created is the mutated ones, the number of them is given by the parameter *Mutation*. Those are selected sequentially from the Parents matrix, and each of the products of each individual has a change of being mutated given by the parameter *Mutation_chance*. For each product to be mutated

the new choice is a random element of the corresponding database.

The remaining individuals necessary to fill the population are new random individuals. The next generation is then complete and the algorithm advances to the next iteration.

It would be possible, and likely if no restrictions are in place, that after a number of iterations the individuals in the *Parents* matrix become very similar and eventually repeated. To avoid this, a test was included to check whether there are repeated individuals in the next generation and mutates everyone found. The test runs only on the part of the generation that was created by crossing the parents without mutation and consists on checking all the possible pairs. The test does not consider the individuals created by mutation and those randomly generated. While it is still possible that they are repeated, the probability is low, not worth the cost of testing them, and they will not persist in following generations as they will either be lost or become parents, in which case their children will be tested. The code for the main algorithm as well as the auxiliary functions is included as an appendix at the end of the thesis. The next two subsections describe those auxiliary functions that are called by the main algorithm during execution.

7.1.4 Overall goodness function

The function named *Overall_goodness* converts the goodness on every performance attribute into a single number that reflects the goodness of a particular product. It is a weighted sum with a parameter s to vary the degree of compensation allowed between different attributes, i.e. how a poor score in one attribute can be compensated by a high score in other attribute. The equation is:

$$Overall\_goodness = \left[\frac{\sum_{i=1}^{12} \alpha_i * g_i^s}{\sum_{i=1}^{12} \alpha_i} \right]^{\frac{1}{s}} \tag{7.1}$$

$s > 1$ High degree of compensation

$s = 1$ Normal weighted sum

$s < 1$ Low degree of compensation

Parameter s can be varied to observe how the solutions generated change with it.

7.1.5 Objectives function

The objectives function is a one input two outputs function that takes a proposed product family and returns its score in the two objectives under consideration. The input is a 12 elements vector with the relative position of each product in its own database. The function access each of those databases to fill a 12x6 matrix *eFamily* where each row is one product, the first 5 columns are the components and the 6th column is the overall goodness. The 12 elements vector *Max_number_of_products* gives the maximum number of products intended for production for each application, and the other 12 elements vector *Max_sell_prices* shows the theoretical maximum price at which the product for each application could be sold if the performance for each attribute were ideal, i.e. the overall goodness equals 1. Then a 1-column matrix is created for each component with as many rows as available options for that component. The elements of those matrices are filled with a loop that goes through every product in the family counting the number of the different component options assuming the intended production numbers, e.g. if both trucks for 20 and 25 tons mount the engine #2, and their intended production are 5000 and 7000 units respectively, then the second row of the *Engine* matrix will be 12000.

To calculate the costs of all the components there are three matrices with as many rows as components in each product (5 in this case) and as many columns as the options for the component with the higher number of them. Rows for which the

component has less options than the length of the row can be filled with NA or 0.

- *Base_prices*: minimum prices for each component when the purchased amount is high enough
- *Max_prices*: amount to add to the base prices when purchasing a single component
- *decay_coefficients*: exponential decay coefficient

A 5x1 matrix is created and named *Cost*, where element 1 is the total cost of all the engines for the intended production, element 2 the total cost of all the pumps, etc. One loop for each component calculates those figures going through all the options for each component, calculating the price taking into account the exponential decay function of that component option, and multiplied by the amount of that option in the intended production. The sum of all elements of *Cost* is the objective 2, which will have to be minimized. The objective 1, which unlike objective 2 will have to be maximized, is the sum of each product maximum theoretical price multiplied by the intended production and by the overall goodness. The two objectives are passed back to the main algorithm as a 2 elements vector.

7.2 Results

This section shows the results after running the algorithm for 100000 iterations. The results will be discussed in the chapter *Evaluation and Discussion*.

7.2.1 Choice of parameters

This subsection consists of a list of the values in the algorithm code that can be parametrized. The values given to the parameters are the result of trial and error:

- sPopulation: The number of individuals for each generation was set as 60, or 5 times the number of genes of each candidate family as they appear in the algorithm. Although the real number of variables is 60 itself. There is no method to identify the best population size a priori, as it depends on the particularities of each problem and their searching space, but a typical rule of thumb to start is between 5 and 10 times the number of genes.

- uParents: 40 was selected as the number of individuals in each generation coming as a combination of individuals from the previous generation. This number ensures a good ratio of preservation and combination.

- Mutation: 10 is the number of individuals in each generation that come from mutation of individuals in the previous generation

- Mutation chance: 0.15, this factor ensures the highest probability to have 1, 2 or 3 mutated genes in each individual. the probabilities are shown in figure 7.1. It can be noticed that there is a 14% chance of an element passing to the next generation unchanged, however, as that will happen to approximately only 1 in 7, it is more efficient to keep it rather that introduce a check to spot it and force it to mutate.

- nIterations: The number of iterations or number of generations over which the algorithm will be run. It was set as 100000.

- nRecord: The number of Pareto fronts that are recorded through the searching evolution. This is only for evaluation purposes, and has no effect on the performance. It was set as 50.

The final parameters are summarized in table 7.1.

Figure 7.1 shows the probability of having a number of mutations in a family candidate that goes through the mutation process to the next generation.

sPopulation	6
uParents	40
Mutation	10
Mutation chance	0.15
nIterations	100000

Table 7.1: Algorithm parameters

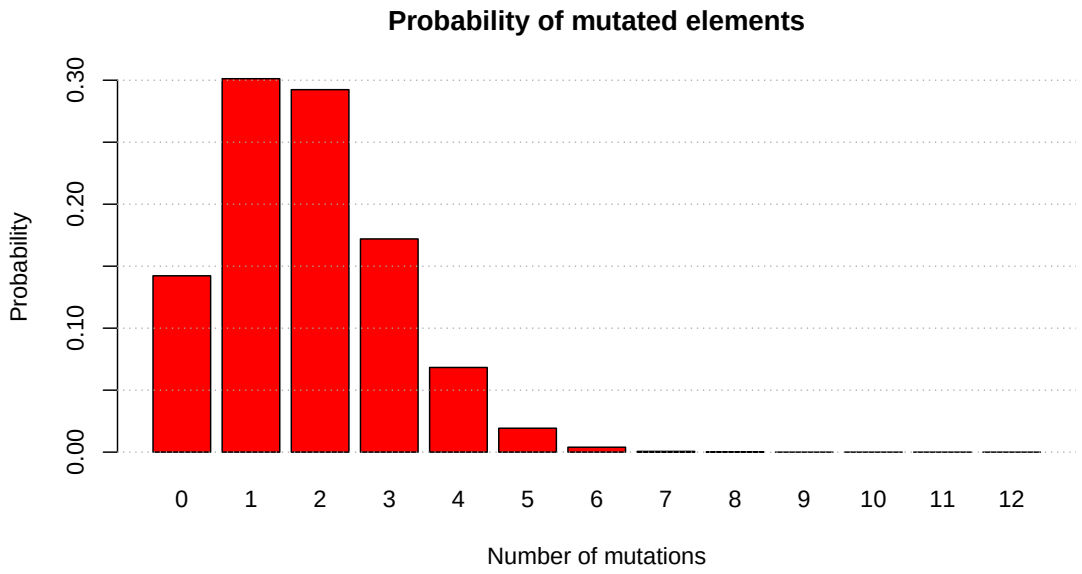


Figure 7.1: Probability of mutating elements

The algorithm was run for 100000 iterations and the historical Pareto front is shown in figure 7.2. It can be observed that the front evolves very little after 25000 iterations. The elapsed time for all the iterations was 3.8 hours, but a very close result was recorded after only 1.4 hours. Another observable feature is that the Pareto points are well distributed over the front, this provides confidence that the algorithm is well balanced and efficient at searching the space, i.e. not focusing on one particular region.

It is important to clarify that the plot in figure 7.2 shows the solution in the objectives space, each point represents the score of the particular solution in both objectives, but does not provide any information about the composition of the candidate family.

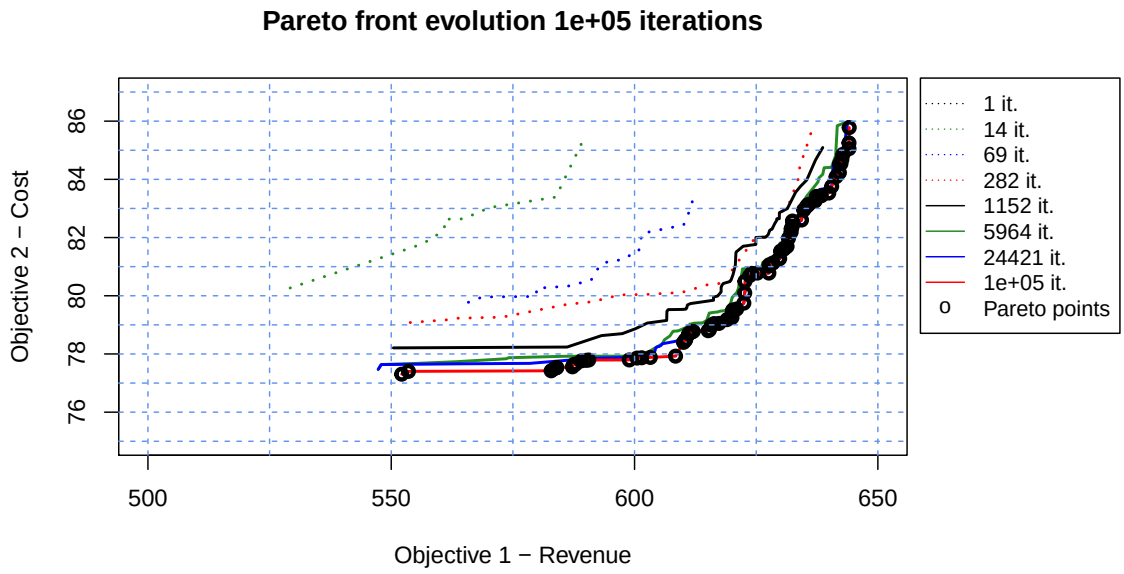


Figure 7.2: Pareto front after 100000 iterations

7.2.2 Improving the Pareto front

To improve the coverage of the Pareto front, an exercise was conducted to find the optimal of each objective separately. That point for the objective 1 - maximum - is easy to find, as it is simply the family with the best performing individual for each application, without any combinatorial involved. On the other hand, finding the optimal for objective 2 - minimum - required a modification of the algorithm to consider only the objective 2 of the fitness function and treat the search as a single objective optimization. That algorithm was run for 10000 iterations and the points are plotted in figure 7.3. The run took 25 minutes. The utopia point is the combination of the best of both objectives, a normally unachievable point that serves as a reference point against which the found solutions can be evaluated, as was explained in the literature review.

Then the main algorithm was re-run for a further 10000 iterations starting with the existing Pareto front and incorporating the extrema for the two objectives in the initial population to ensure that the final Pareto front extends to both ends. The

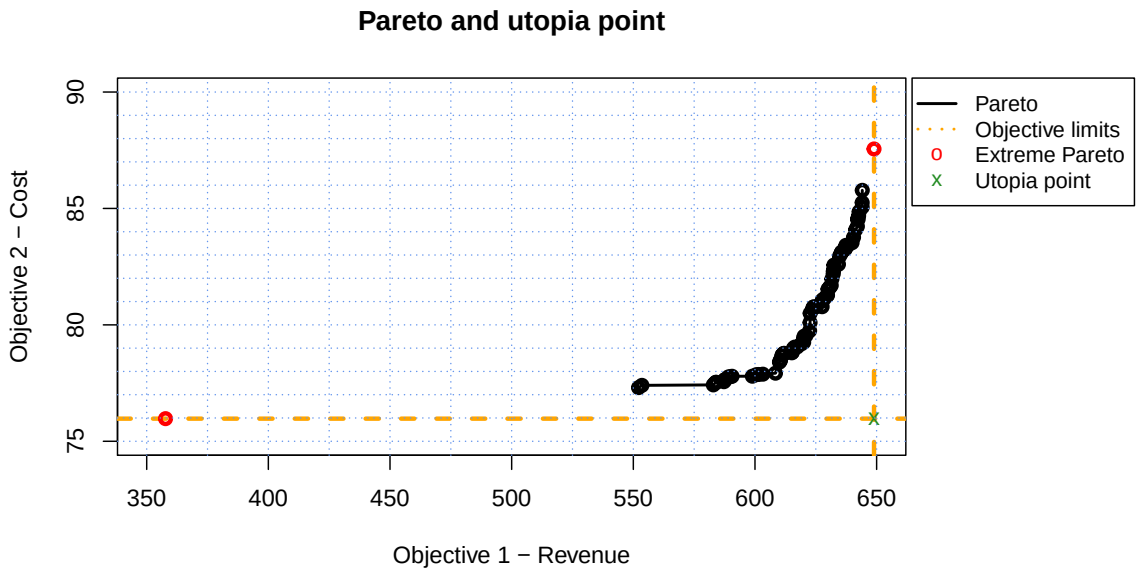


Figure 7.3: Pareto and objectives extrema - utopia point

run took 34 minutes and the new Pareto front is shown in figure 7.4

7.2.3 Commonality

A commonality index was calculated for all the families in the Pareto front. The index is described by Martin and Ishii (1996) and the definition is:

$$CI = \frac{u}{\sum_{j=1}^{v_n} p_j} \tag{7.2}$$

Where u is the number of different parts in each family, p_j is each product, and v_n the number of products in the family.

The higher the degree of commonality the lower the index CI is. The maximum theoretical index for this case study is 0.683 if all the possible components are used at least for one product, and the minimum 0.083 if all the products shared all the components. For the families in the Pareto set the indices range from 0.300 to 0.467, and figure 7.5 shows the index for the different solutions in the Pareto front ordered

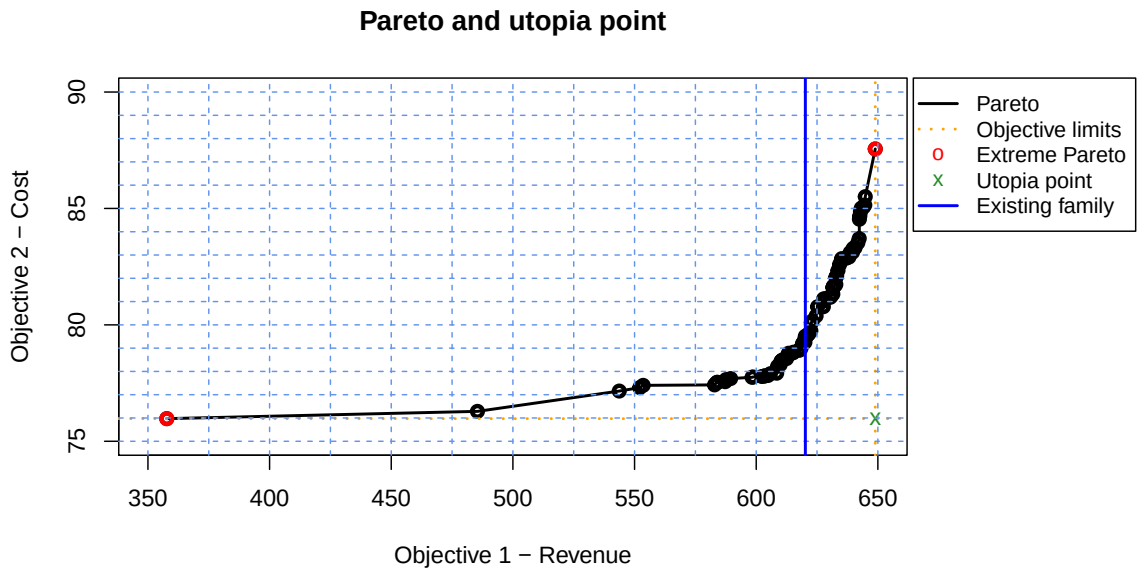


Figure 7.4: Pareto and objectives extrema - utopia point

with growing objective 1, the performance related objective. There is a correlation coefficient of 0.60 and the best second degree polynomial is shown in red. It is clearly seen in the plot that commonality tends to decrease for bigger scores in objective 1, which is to be expected as higher performance products can usually be achieved by designing them more specifically. However, this relation still allows the existence of relatively high degrees of commonality near the top end of highest performance families.

Figure 7.6 shows the prevalence of different products for each application in the Pareto front. The x-axis shows the applications and the y-axis the number allocated to each possible product in its corresponding candidate product database. Bigger circles mean that the particular product appears more frequently in the Pareto solutions. However, the number only represents the position of a product in a database, and does not provide specific information about the composition of that product. This graph only shows the relative variety of products for each application but without any reference to commonality, two distinct circles may equally refer to two products almost identical or completely different. Figure 7.7 shows the same in-

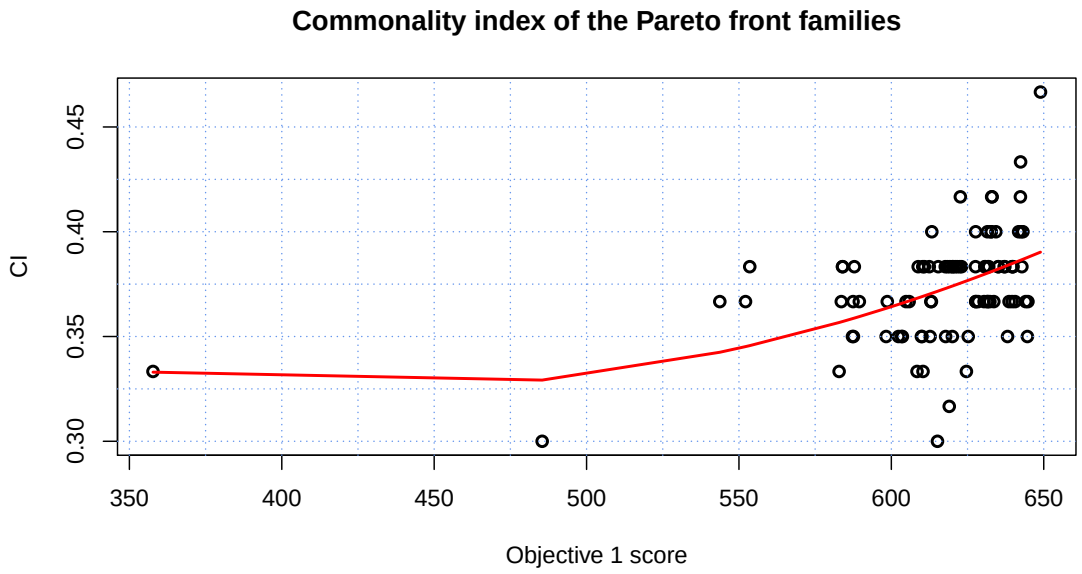


Figure 7.5: Commonality index of the Pareto front families

formation as figure 7.6 but without referencing each product to the position they occupy in the databases. In this plot, lower reaching columns mean that the solutions for that particular application tend to gravitate towards a reduced number of products, whereas higher columns mean that the preferred options for that application are more varied. In that figure, the grey filled circles show the prevalence of each product and are flanked by a series of red circles to the left showing the prevalence of those product in the half of the Pareto set with the poorest scores in the revenue objective and best scores in the cost objective; and a series of green circles to the right corresponding to the half of the Pareto set with the best scores in the revenue objective and poorest scores in the cost objective. It can be observed that some products have only one of the side circles, and therefore they only appear in one of the halves of the Pareto front. This can be made clearer in figures 7.8 to 7.12, which show the prevalence of each component option for each application with the same convention of a grey filled circle for the total, red circle on the left for those solutions with better cost objective and green circles for those with better revenue objective. All the products for applications t20 and t25 with the engine #2

appear on the right hand side of the Pareto front, which can be expected as this is a more powerful engine and is favoured in solutions with better performance. Another interesting fact is that all the solutions use engine #4 for all the applications beyond t50, whereas for applications t35, t40 and t45 most solutions employ engine #4 but some use engine #3, and are located in the area of the Pareto front where the cost objective is stronger. It can be observed that engines #5 and #6 are unused in the entire set of Pareto solutions.

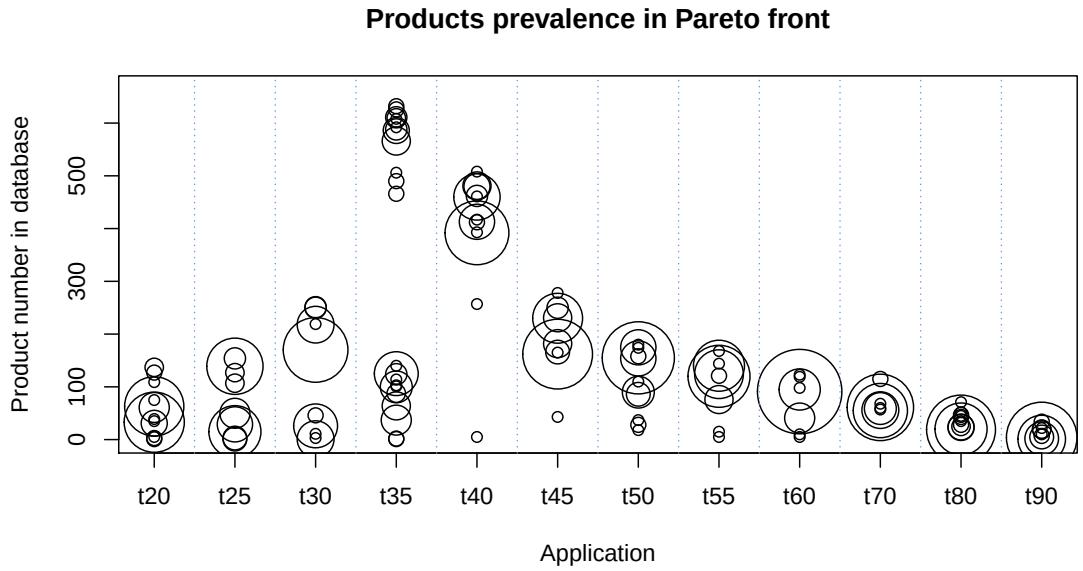


Figure 7.6: Product prevalence in Pareto front given their positions in the databases

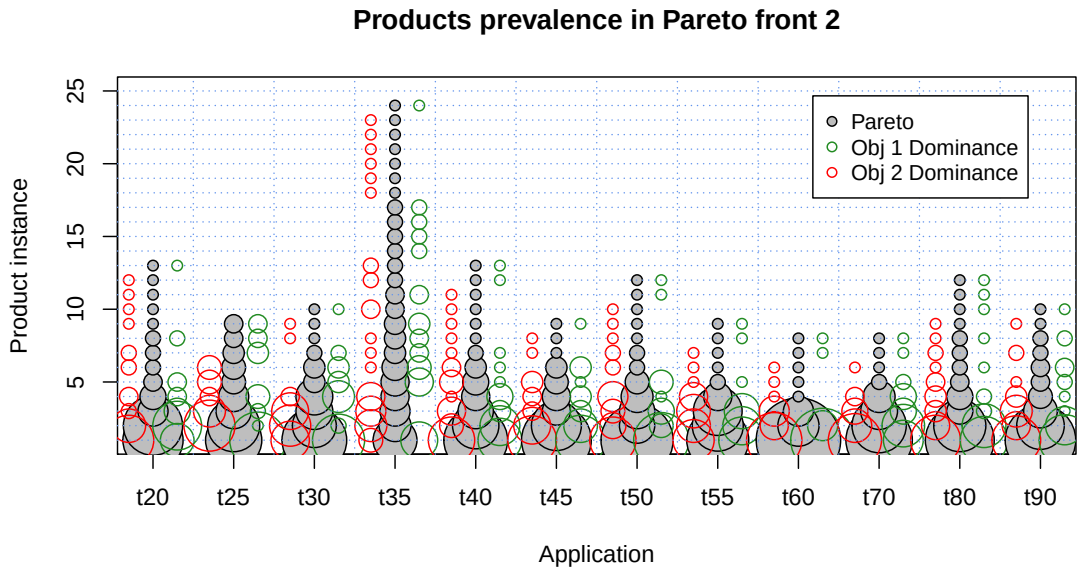


Figure 7.7: Product prevalence in Pareto front irrespective of their positions in the databases

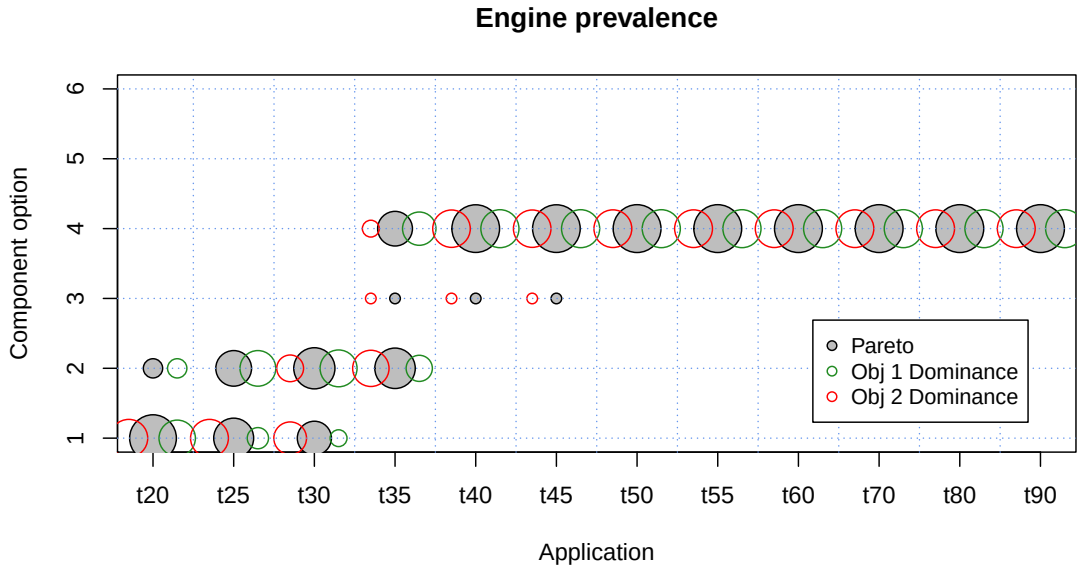


Figure 7.8: Engine prevalence in Pareto front

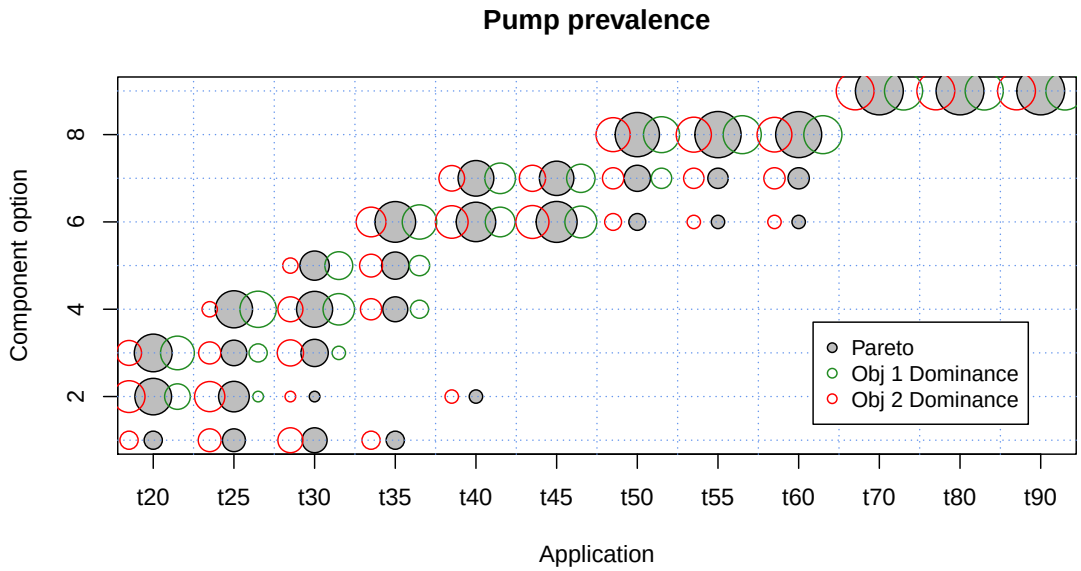


Figure 7.9: Pump prevalence in Pareto front

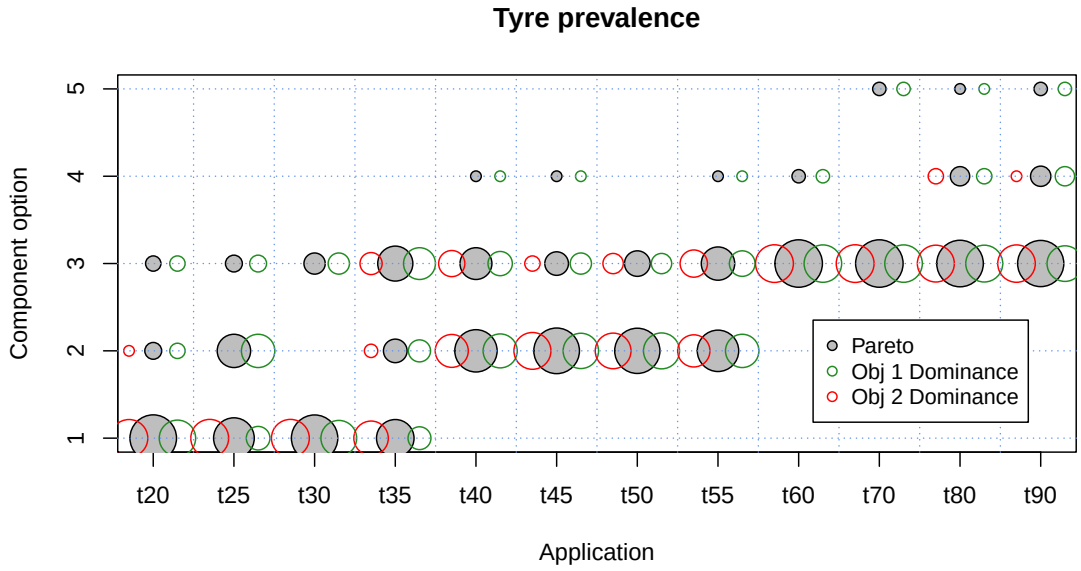


Figure 7.10: Tyres prevalence in Pareto front

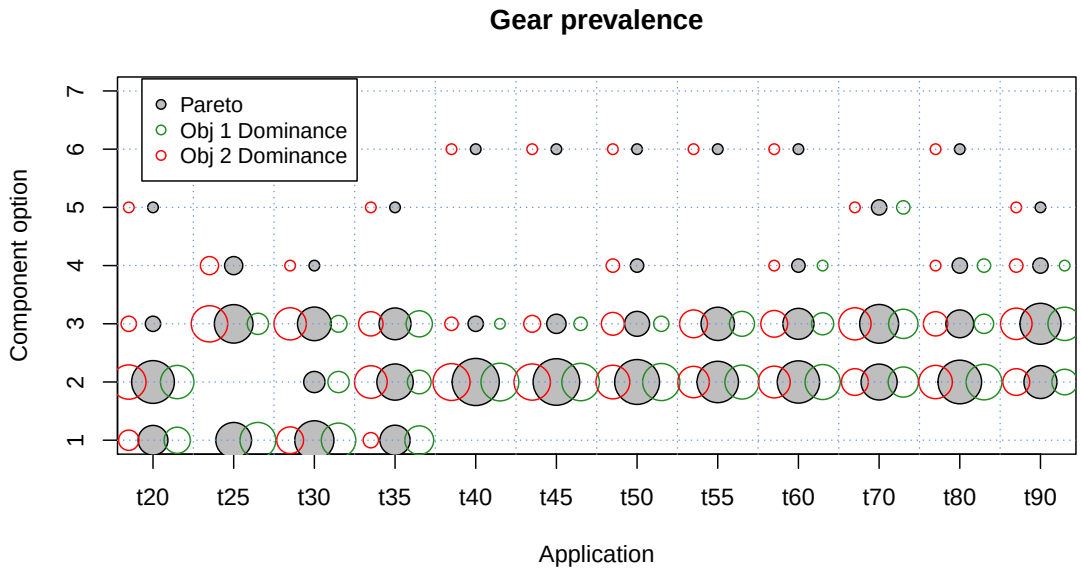


Figure 7.11: Gears prevalence in Pareto front

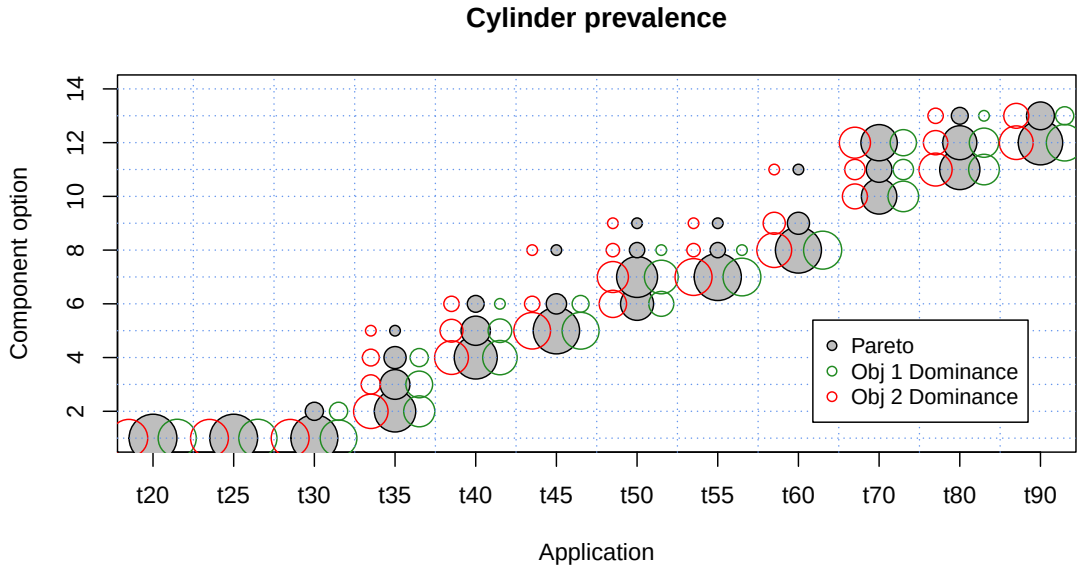


Figure 7.12: Cylinder prevalence in Pareto front

7.3 Number of products

One of the main characteristics of this method is that the input is the applications to be covered, rather than the products of the family. Table 7.2 shows the distribution of the Pareto solutions according to the number of distinct products. the table reads as:

- Column 1 (Distinct): Number of products composing the family
- Column 2 (Products): Number of Pareto solutions with the number of distinct products specified in column 1
- Column 3 (Prod - Cyl): Same as column 2 but ignoring the cylinder to consider the products equal
- Column 4 (Drivetrain): Same as column 2 but considering only the drive train, i.e. engine, gearbox and tyres
- Column 5 (Engine/pump): Same as column 2 but considering only the engine and the pump

Table 7.2: Solutions by number of products

Distinct	Products	Prod - Cyl	Drivetrain	Engine/pump
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	2
5	0	0	5	20
6	0	0	23	42
7	0	0	37	23
8	0	9	20	4
9	1	38	6	0
10	18	32	0	0
11	44	11	0	0
12	28	1	0	0

It is observable that most Pareto solutions do not have a one to one map of products to applications.

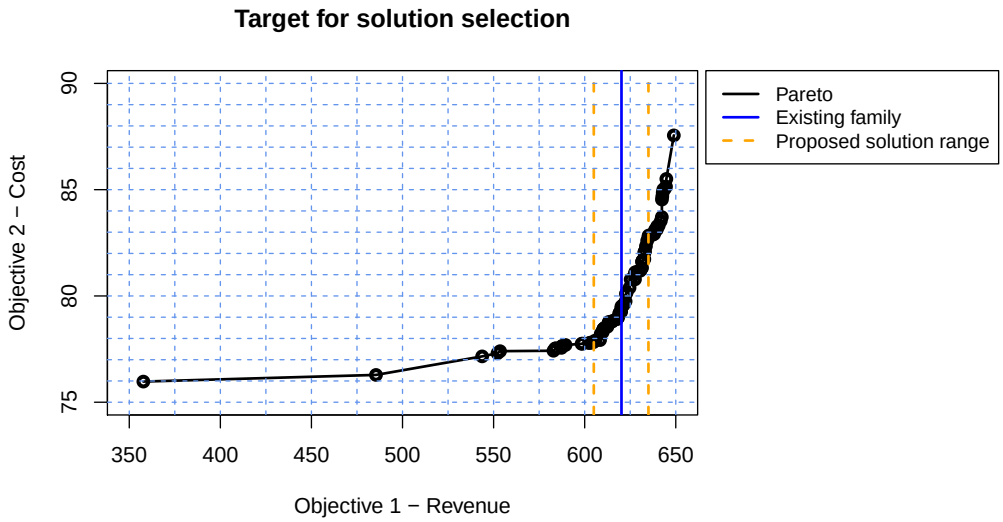
7.4 Solution selection

The Pareto front still has too many solutions for the decision maker to chose one. Although not part of the target of this case study, this section will show a possible way to filter some solutions so that the decision maker is presented a feasible problem. Figure 7.13 shows the Pareto front and the value for the revenue objective of an existing product family with a blue line. If the target is to replace or compete against that family, the suggested area where the solution can be chosen from can be delimited by the orange dashed lines. In case the target were to move up the market from that existing family, then the solutions should be picked from the region strictly to the right of the blue line.

In the first case, the selection still shows 16 potential solutions. A possibility is to consider the results shown in table 7.2 and assume that solutions with a reduced number of different products will have an advantage in terms of development time, apart from the parts cost reduction which is already computed. This is shown in table 7.3.

The table shows four different criteria to count the number of different products that compose the family. Each of those criteria consider that two products are equal when:

- All components are the same
- All components are the same except the cylinder
- The components of the drivetrain are the same
- The engine and the pump are the same



choice.pdf

Figure 7.13: Pareto front and proposed region for choosing the final solution

Table 7.3: Solution subset showing number of products under several criteria

Obj1	Obj2	Products	Prod - Cyl	Drivetrain	Engine/pump
615.1304	78.79901	10	9	7	4
615.3839	78.82706	12	11	8	6
617.6872	78.90973	12	10	8	6
617.8095	78.92783	11	9	7	6
618.4083	**78.99226**	**12**	**8**	**6**	**6**
618.9968	79.17926	10	9	7	4
619.1906	79.21132	12	9	7	7
620.0175	79.25061	11	9	8	5
620.0959	79.42224	11	10	6	6
620.1865	79.51168	11	10	9	5
620.9794	**79.54349**	**10**	**8**	**6**	**7**
621.646	79.61953	10	9	7	6
622.4619	**79.7469**	**10**	**8**	**6**	**6**
622.6426	**80.09755**	**10**	**8**	**6**	**6**
623.0333	80.16494	12	10	9	6
624.6233	80.38533	9	9	7	5

Of those four criteria, the most reasonable to use is the second one, as the cylinder is a customized component with little difference in terms of development time and effort. According to that criterion, the most suitable solutions are those in bold typeface, and table 7.4 shows the truck numbers for each application.

Table 7.4: Trucks in the selected solutions

t20	**t25**	**t30**	**t35**	**t40**	**t45**	**t50**	**t55**	**t60**	**t70**	**t80**	**t90**
33	15	170	125	392	162	155	140	91	55	19	1
33	50	170	125	460	230	155	120	91	56	19	1
61	50	46	566	460	230	86	120	91	56	19	4
61	50	46	566	460	230	86	120	91	56	19	22

Figure 7.14 shows the map between the original applications and components for one of the proposed solutions as described in the figure 1.2 in the introduction chapter.

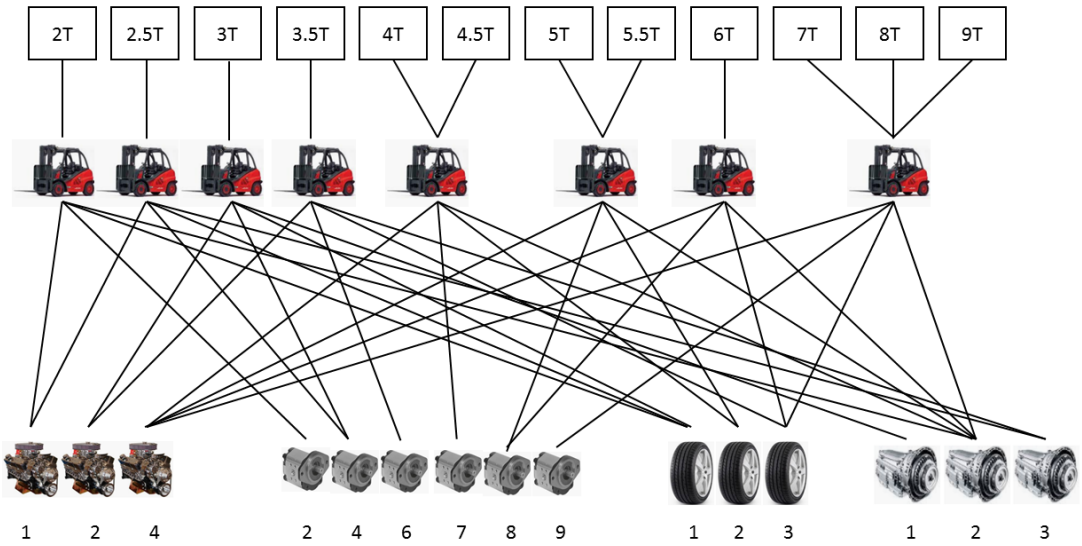


Figure 7.14: Applications to components map for one of the proposed solutions

7.5 Existing products

An alternative scenario is that one in which some products already exist and the problem is how to fit the remaining products around it. For this case, the existing product can be modelled and fixed while the search looks to optimize the whole family with the constraint that one or more of those products are already defined. The results are expected to be different and take into account the existing conditions. To test this hypothesis, it was assumed that the product for the 8 tonnes application is already existing and it mounts engine #5. This engine does not appear in any of the solutions found in the original run, however, with the condition that it is already part of the family, this now appears in some solutions for a product other than the one that is fixed with it as it can be seen in figure 7.15. This happens thanks to the cost reduction associated with increasing the number of those engines.

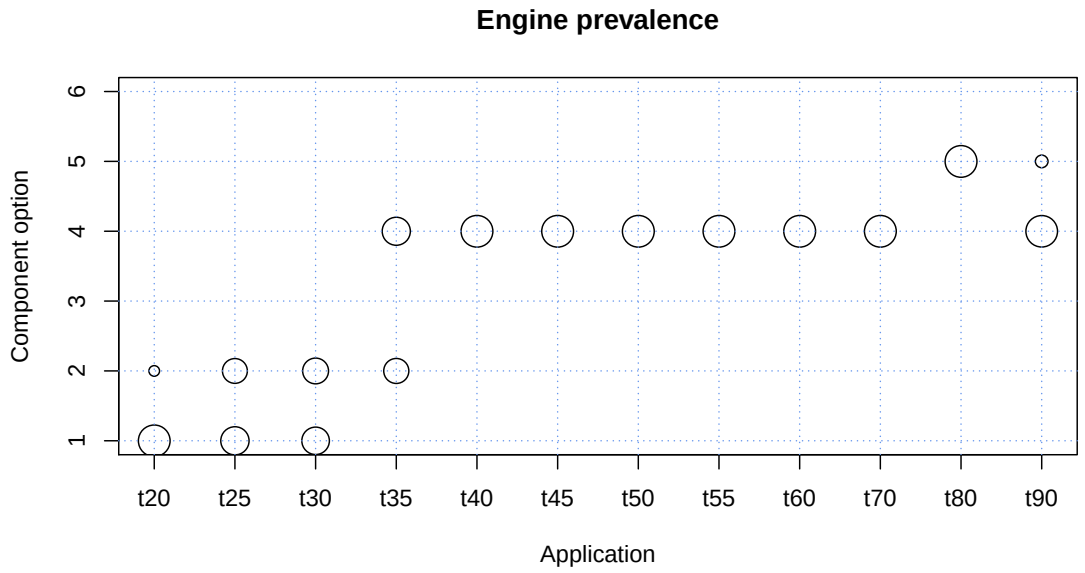


Figure 7.15: Engine prevalence with a product already fixed

This chapter has shown the results for the case study and an analysis of those results. the following chapter will provide a series of validation tests that use the Pareto front obtained from this chapter as the main basis.

Chapter 8

Validation

This chapter describes all the steps that were taken to validate the results of the case study. Due to the problem nature, it is not possible to prove the results with the rigour that would be required for a mathematical theorem. Instead, the validation exercise consists of a set of tests intended to increase the confidence in all the steps involved. The tests defined are:

- Performance model validation: comparison of test results for a known product with the simulation results for that same product.

- Objectives validation: an exercise involving the manual ordering of several products and families according to expert's preferences and comparison with the model ordering.

- Searching algorithm validation: analysis of the genetic algorithm performance.

- Solution analysis: analysis of the Pareto solutions against sanity tests designed by engineers with expertise in the field.

8.1 Performance model validation

The models were parametrized to replicate the characteristics of three existing products and the results of the simulation compared with the declared specifications datasheet for those products. The differences are summarized in the table 8.1, and the comparison for each product represented in the spider diagrams in figures 8.1 to 8.3. The red lines are the declared specifications and the black lines the simulation data. The subsequent subsections discuss the accuracy and its significance.

Table 8.1: Performance model validation results

	2.5 Tonnes			3.0 Tonnes			3.5 Tonnes		
	Specs	**Sim**	**Diff (%)**	**Specs**	**Sim**	**Diff (%)**	**Specs**	**Sim**	**Diff (%)**
Fuel (l/h)	3.1	3.2	2.2	3.5	3.5	-1.0	4.0	3.9	-3.0
SpeedU (km/h)	17.5	17.8	1.6	18.9	19.3	2.0	16.6	16.7	0.4
SpeedL (km/h)	17.3	17.2	-0.4	18.7	18.7	-0.2	16.2	16.1	-0.4
LiftU (m/s)	0.69	0.71	2.3	0.61	0.61	-0.1	0.56	0.58	4.4
LiftL (m/s)	0.66	0.66	-0.1	0.58	0.57	-2.1	0.52	0.54	4.7
DrawU (N)	12100	12434	2.8	13800	13929	0.9	14000	14924	6.6
DrawL (N)	14800	20357	37.5	13400	19029	42.0	17500	21461	22.6
GradU	0.31	0.31	2.4	0.31	0.31	3.1	0.28	0.31	12.3
GradL	0.15	0.32	106.4	0.12	0.25	101.7	0.15	0.25	67.4

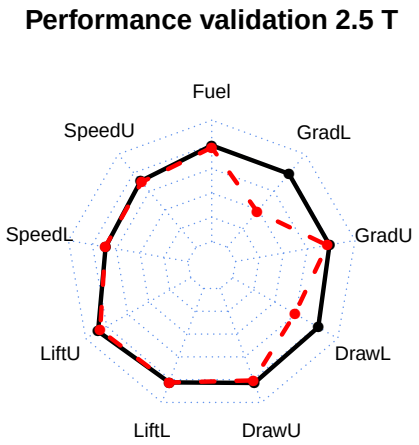


Figure 8.1: Performance validation 2 & 2.5 T

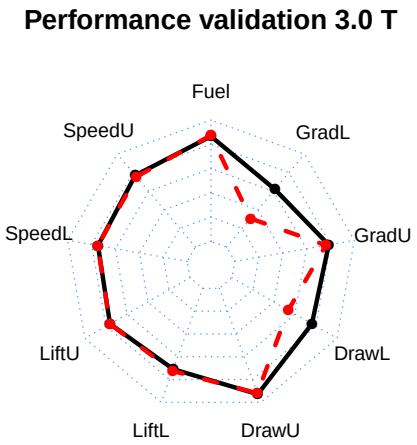


Figure 8.2: Performance validation 3 & 3.5 T

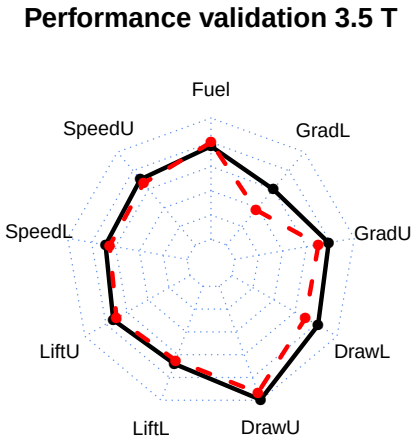


Figure 8.3: Performance validation 4 & 4.5 T

8.1.1 Fuel consumption

The differences between simulation and specification are negligible, less than 5% in all cases, which is lower than the differences that can be encountered between two actual tests. Real tests involve human factors, no two drivers will get the same results, and in addition, environmental conditions such as atmospheric pressure, altitude, temperature, etc. have an impact on the figures that can add to 10% in practice.

8.1.2 Maximum speed

The differences are minimal and acceptable according to engineering criteria. The achieved figure in the tests is also dependable on environment conditions, to which for this case should also be added the state of the tyres and the characteristics of the test track and its effects on the rolling resistance. The biggest source of error in general for maximum speed prediction for vehicles is the aerodynamic drag, and this is not relevant for the speeds involved in this case.

8.1.3 Lifting speed

The most important potential source of error for this attribute is the characterization of the hydraulic system and the pressure losses model. In this case, the errors are contained within 5%, which makes them acceptable.

8.1.4 Drawbar pull

These results seem to be accurate for unladen cases, within 7%, and with a bigger margin for laden cases.

8.1.5 Gradeability

The results are similar to those for drawbar pull, acceptable for the unladen case where the higher deviation is 12.3% and seemingly too high for the laden case. Although the 12.3% maximum error may seem to be problematic, small differences in the friction coefficient in the test and model can have a big effect in reported values as it may make the truck skid despite still having enough torque available.

8.1.6 Addressing inaccuracy

Considering the issue with laden gradeability and drawbar pull, the objectives function was modified to ignore those specific attributes and the searching algorithm run again. A new Pareto front was found with these new settings and the components of its solutions compared with those in the original solution. Then the original objectives were calculated for the Pareto solutions found with the modified version. This is shown in figure 8.4 and it is clearly observable that they do not match the original Pareto front. This is expected, as the new solutions are optimized ignoring two attributes, it would be a coincidence that they would match the solutions found with a function that incorporates those attributes.

Figure 8.4 shows the Pareto points found with the alternative function and how they perform in the original objectives compared to the original Pareto front.

Figure 8.5 shows the rate of occurrence of particular product candidates in the original Pareto and the occurrence of the same candidates in the alternative version. The different applications are separated by dark blue solid lines. Horizontal lines where the black dot is too low or zero and the red cross is high, or vice versa indicate that this particular product candidate is common in one set of solutions and not in the other one. It is observable that there are many instances of that.

An additional analysis performed was to consider the individual components of every family in the new Pareto front, find the family in the original Pareto front

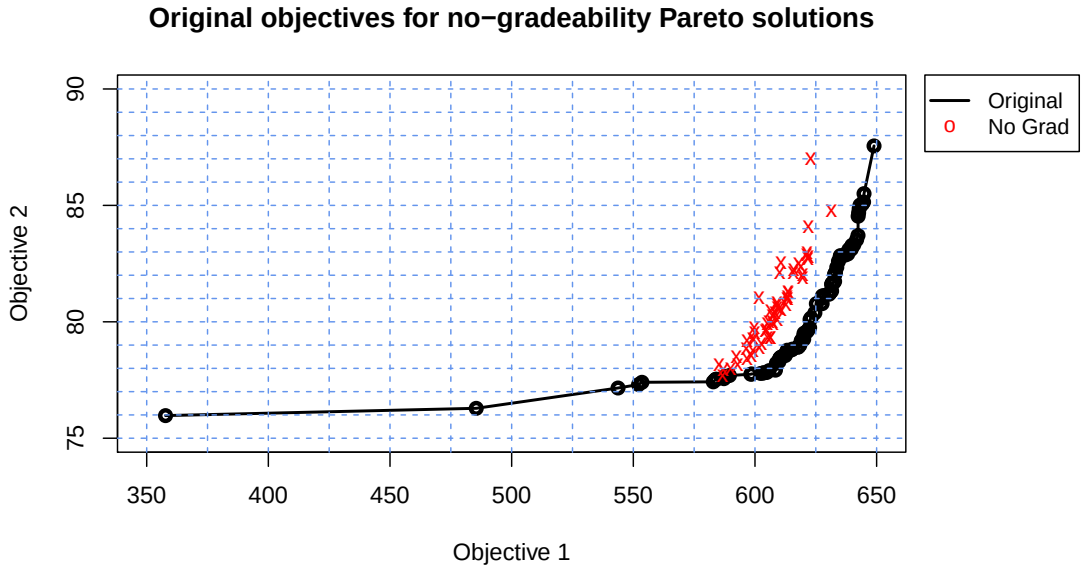


Figure 8.4: Original objectives for the solutions found disregarding the performance attributes with less confidence

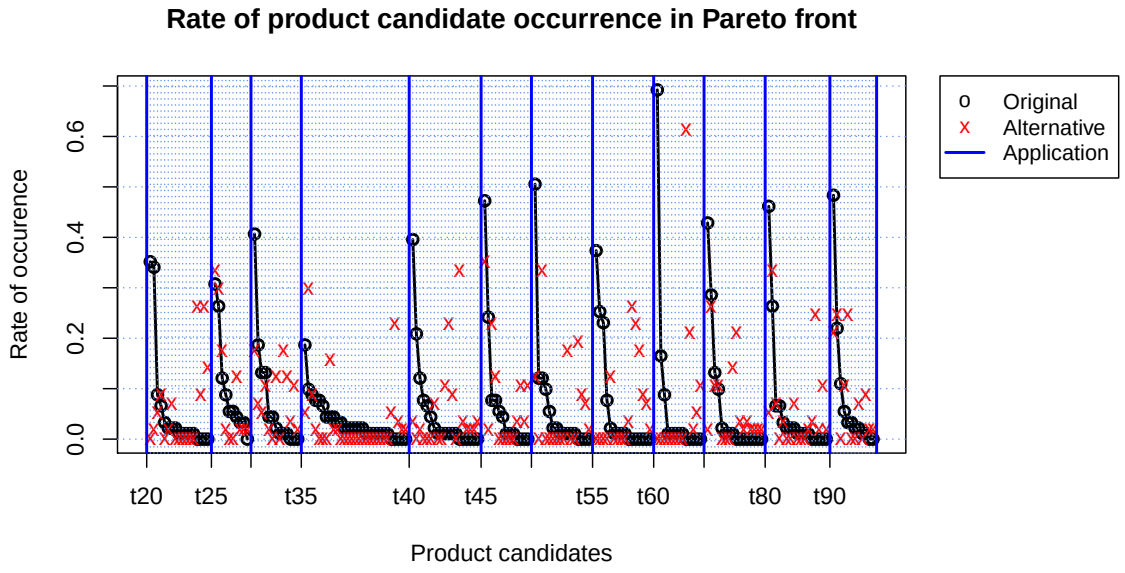


Figure 8.5: Rate of occurrence of product candidates in the original Pareto vs alternative Pareto

that most closely matches it and count the number of equal components. The total number of components in a family is 60. None of the families in the alternative Pareto set was found to be repeated in the original Pareto set, and the number of components common with a family in the original Pareto ranged between 43 and 51. Figure 8.6 shows the distribution.

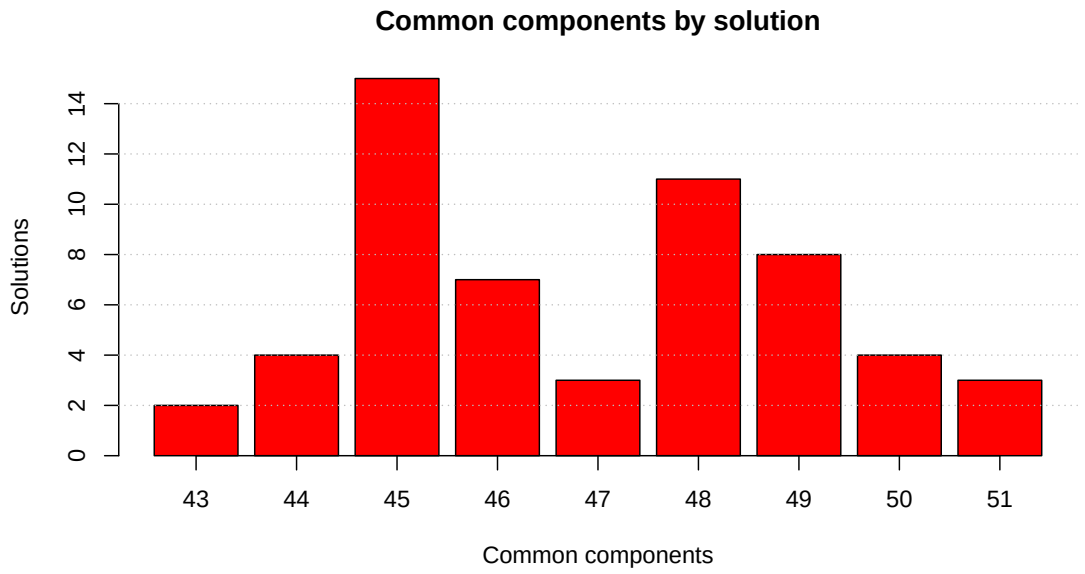


Figure 8.6: Closest match in original Pareto front by number of components

8.1.7 Conclusion

The accuracy of the simulation models with respect to the declared figures is high with the exception of drawbar pull laden and gradeability laden. The reason for inaccuracy in those attributes is the high sensitivity to small variations in the friction coefficient which is difficult to control in testing. However, the models should predict accurately the differences between two product candidates. It has also been shown that the accuracy of these models is of critical importance for the whole method as it can vary the solutions even in the case that the particular attribute has a low relative importance.

8.2 Objectives validation

Two sets of 10 product samples were selected, the first one for 3 tonnes and the second one for 6 tonnes. The sets were given to experienced engineers and marketing professionals and they were asked to order them from best to worst. The only information they were presented was the performance specifications in the nine attributes considered, no information about the components used for each example was included. After ordering them, they were asked to go through the examples in pairs, with the one that was ranked higher on top, and provide a measure of confidence in the adequateness of the order:

- 3 - Sure the top product is better
- 2 - The top product is better but with doubts
- 1 - Not sure which product is better

Then the level of confidence for each pair was assessed against the difference in the overall goodness between those products provided by the model, the results are plotted in figure 8.7, the y-axis does not have any meaning, each point is plotted at a random height for the sake of visual clarity and avoid overcrowding. It can be appreciated that the majority of pairs with confidence level 3 corresponds with positive differences in goodness, i.e. the model agrees with the response, whereas rankings with lower level of confidence, or no confidence at all are located more towards the 0 axis or to the left of it. A green cross to the left of the 0 axis means that the respondent ranked that pair in the opposite direction with respect to the model, but were not sure at all that their ranking was correct. It cannot be expected that there would be a complete agreement between the responses and the model ordering, as the orderings made by different people differ in some cases. This is normal as there is not a single and undisputed way in which every decision agent will incline towards one side or the other in front of a tight trade off. This is the

main reason why this thesis focuses on a multi-objective approach with a posteriori articulation of preferences.

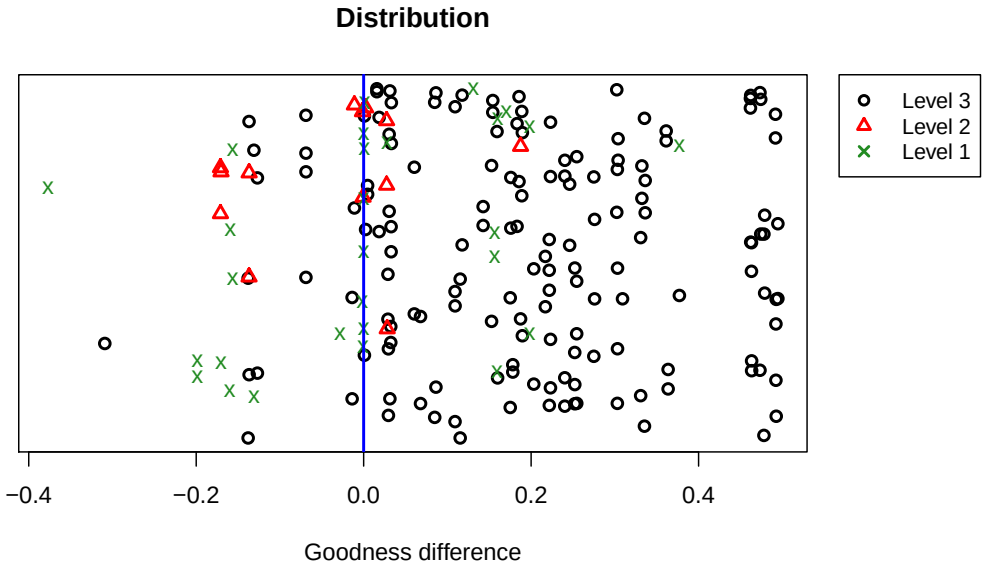


Figure 8.7: Confidence levels vs goodness difference

A similar exercise was attempted for the entire objective 1, i.e. evaluating families, however, this was not successful as the respondents were not capable of comparing two families and articulating their preferences when the trade-off involved 108 different attributes, except for cases where one family dominated the second one in most of the objectives, which is not the general case among families belonging to the Pareto front.

8.2.1 Conclusion

This test has shown a correlation between the order in which the model ranks a number of sample products and the preferences of professionals with knowledge and experience in the field. This ordering, or the assignment of a well-ordered score to every product candidate is an important part of the method described in this thesis, as it is the benchmark against which one of the two objectives of the problem is

assessed, and hence the solutions chosen.

8.3 Search algorithm validation

8.3.1 Statistical test

A measure of distance to Pareto was defined as the distance from that solution to the best values of the two objectives found in all the Pareto points that dominate the given solution. Figure 8.8 shows an example solution compared to the Pareto front. The red dotted lines mark the subset of the Pareto front that dominates that sample solution, and *dObj1* and *dObj2* are the distances to consider. The smaller those distances, the closer the solution will be to the Pareto front, and if the distances happened to be negative, that would mean that the given solution dominates a subset of the Pareto front. In that case, the solution would be a new Pareto point that would replace all those that it dominates.

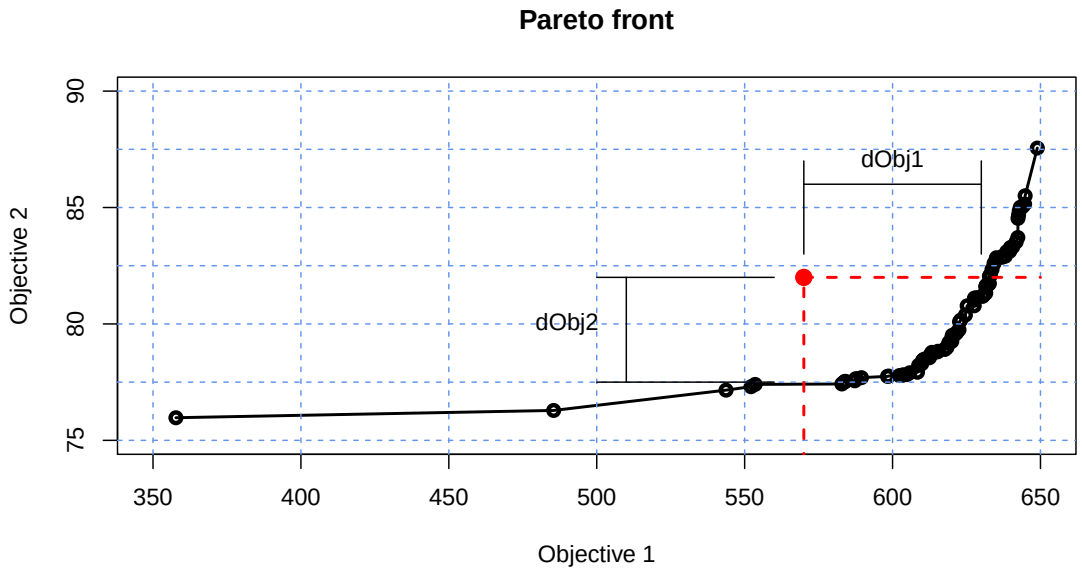


Figure 8.8: Distance from a generic solution to the Pareto front

A random search was conducted to find 100000 solutions, and the distances

dObj1 and *dObj2* were computed for all of them. Table 8.2 shows the maximum, minimum, average and standard deviation of the distances to the Pareto front for that solutions sample set.

Table 8.2: Random solutions statistics

	dObj1	dObj2
Max	349.16	17.26
Min	65.27	3.41
Avg	200.59	10.91
SD	35.36	1.89

The next step was plotting the distances probability density and compare it to a normal distribution using the data of table 8.2 as parameters. This is shown in figures 8.9 and 8.10 respectively for the objectives 1 and 2. The black dots are the sample points and the red lines are the normal distributions calculated with the mean and standard deviation taken from those samples.

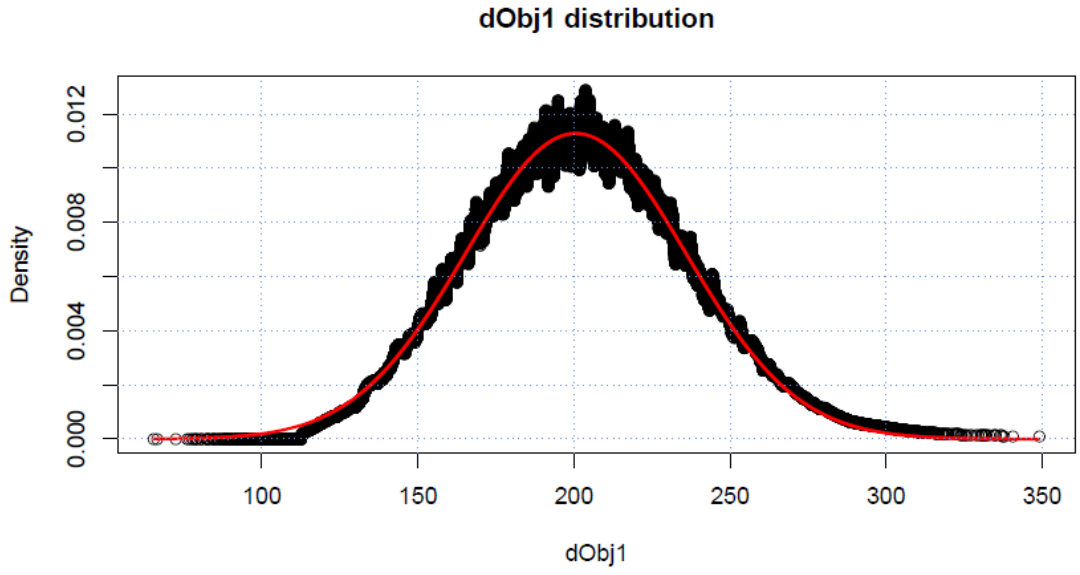


Figure 8.9: Solution sample set dObj1 distribution

It appears clear that the distances to the Pareto front closely follow normal distributions. Assuming this to be true, a statistical analysis follows:

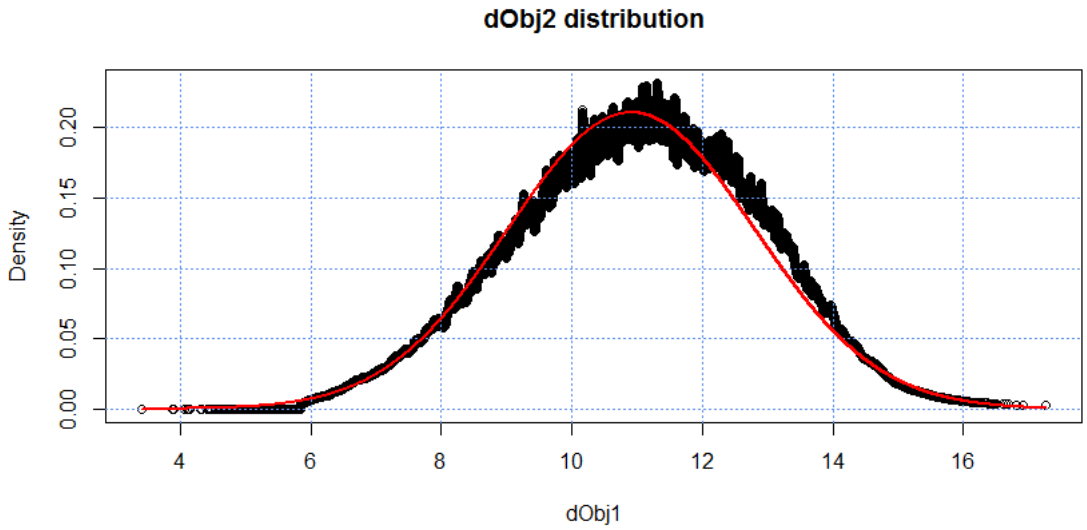


Figure 8.10: Solution sample set dObj2 distribution

Any point of the Pareto front will have distances dObj1 and dObj2 = 0, this is 5.67 and 5.76 standard deviations below the mean. Assuming the normal distribution rule holds for all the possible solutions, this implies a cumulative distribution function value of 0.999999993 and 0.999999996 (Six Sigma Institute, 2019), and a probability of randomly finding one of between 1 in $1.43 * 10^8$ and $2.5 * 10^8$. Since the algorithm was run for 10^5 iterations and the number of candidates was 60 per iteration, a total of $6 * 10^6$ families were evaluated. With this number of evaluations, the probability of randomly finding at least one of the Pareto solutions in the range found by the algorithm is 4.1% or approximately 1 in 24, and that of finding a front with 91 elements falls to nearly zero, in the range of $7.7 * 10^{-125}$%, or 1 in $1.3 * 10^{126}$.

8.3.2 Pareto convergence

The algorithm was run two additional times with the same number of iterations and the Pareto fronts compared. Figure 8.11 shows that the shape and position of the three fronts is equivalent, which proves that the algorithm is consistently finding points of that quality.

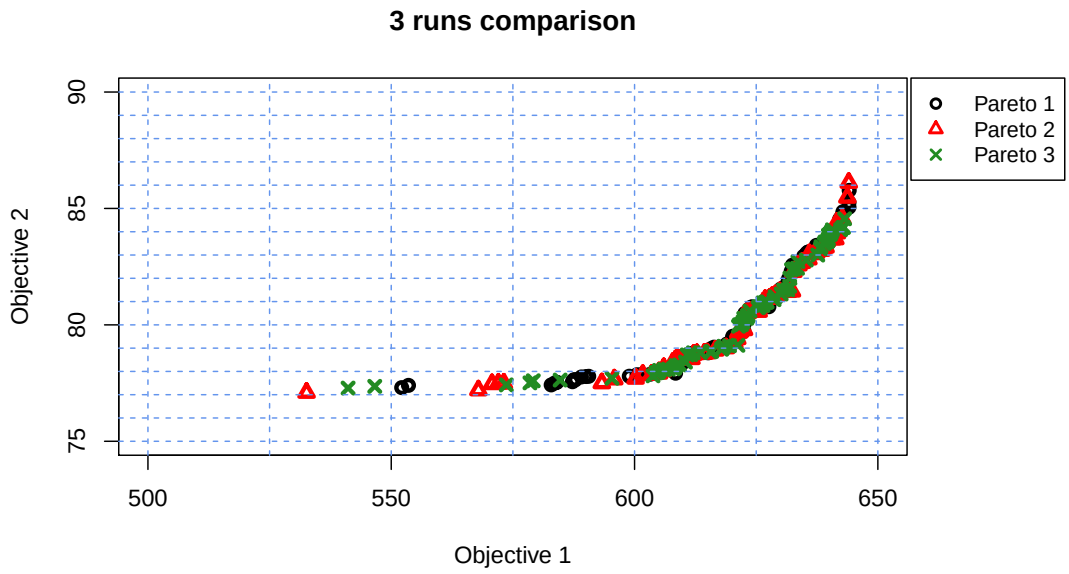


Figure 8.11: Pareto comparison from 3 different runs

8.3.3 Conclusion

This test has compared the efficiency of the search algorithm with that of a random search and bounded the improvement assuming a normal distribution is applicable for all the searching space. Although it is not possible to know how much further the absolute Pareto front could be, as the normal distribution law can break at that kind of distance from the mean, it can be concluded that the searching algorithm is efficient at finding a reasonably good set of solutions. Genetic algorithms are heuristic methods for which finding a good solution is the target, they cannot find exact solutions.

8.4 Industry designed tests

A set of tests was designed in collaboration with engineers with expertise in the field as a sanity check of the families belonging to the Pareto front. Success in these tests is intended to increase the confidence in the results, whereas failure of the tests

would point to weakness and cast doubts on the validity of the method.

8.4.1 Dislocations

A dislocation is a case in which, within the same family, a product intended for a lower rated capacity than other product mounts a bigger engine, pump or cylinder. This is not a standard term, but an ad hoc definition for this validation exercise. Intuitively, it is hard to find sense in these dislocations happening, a potential customer would find it weird that a higher capacity product incorporates a smaller engine for example. Their existence is an unexpected result from the logical point of view, but not so much from a purely algorithmic point of view. This would not make sense in the real world.

Procedure

49 out of the 91 families in the Pareto set showed at least one dislocation. Table 8.3 shows the distribution of families by number of dislocations, whereas table 8.4 shows the occurrence per component. It is observed that the rarest dislocation is that one of an engine.

Table 8.3: Distribution of families by number of dislocations

Dislocations	Instances	%
None	42	46.1
1	37	40.6
2	10	11
3	1	1.1
4	1	1.1
5 or more	0	0

After finding these dislocations, a script was written to correct the families in which they occurred, and check whether the corrected family would dominate the current one. Two methods were defined to correct the dislocations:

Table 8.4: Distribution of Components by number of dislocations

Component	Instances
Engine	2
Pump	37
Cylinder	25

1. Going from lowest to highest capacity and upgrading any case of component lower than the one in the previous product.

2. Going from highest to lowest capacity and downgrading any case of component higher than the one in the previous product.

For each family showing dislocations, all the combinations of the two methods for each component with a dislocation were checked. A total of 117 families were generated this way.

18 of the 49 families with dislocations were dominated by at least one of the corrections done and were replaced in the Pareto front. Including those 18 families, a total of 94 of the corrected families were non dominated by any of the existing Pareto front, and hence worth to be added to that Pareto front. This represented 80.3% of the corrected families. After that, it was checked whether the new additions to the Pareto front made any of its existing or new solutions dominated, the Pareto front was finally reduced to 88 elements.

Only 7 solutions with dislocations remained in the Pareto front at this point. Figure 8.12 shows a zoom of the Pareto front showing those solutions in black and the neighbouring Pareto solutions in red. The dotted line is the linear fit between the neighbouring solutions, and the limit for any intermediate solution for a convex Pareto front, i.e. if the Pareto front is convex and continuous, any point behind that dotted line cannot be a true Pareto solution, and although not found by the algorithm, there exists a solution that dominates it. Due to the discrete character of this problem, it cannot be assumed that the Pareto front is convex at every point, and it is definitely not continuous, and hence it cannot be ruled out that those

solutions with a dislocation be true Pareto points. In case they are true Pareto points, the reason for they to appear lies with the choice of searching algorithm, as genetic algorithms are capable to find such points, which can not be said of all other possible methods. What can be said, however, is that those points are unlikely to be good choices, as they represent a relatively big sacrifice in one objective for a relatively small benefit in the other one. Normally a decision maker would opt for one of the neighbouring red points rather than the black one.

With the exception of the points plotted in the rows 2 and 4 in the first column, all the other solutions with a dislocation are of the type mentioned in the previous paragraph.

Conclusion

The existence of dislocations in the Pareto solutions after the algorithm run is due to the nature of the problem, and unavoidable without including some intelligence in the algorithm to force a deviation from that kind of solutions.

When a dislocation is introduced over a non-dislocated Pareto solution, it is very likely that one of the two objectives is improved at the expense of the other one, i.e. the dislocated solution is not dominated by the non-dislocated solution and hence both can be Pareto solutions, even though the dislocated one does not appear to be reasonable. This has been an unexpected finding, and it is likely to happen more and more often the higher the number of objectives that compose the fitness function. From an industrial point of view, solutions with a dislocation are not interesting, and should be eliminated from the Pareto set that is presented to the decision maker.

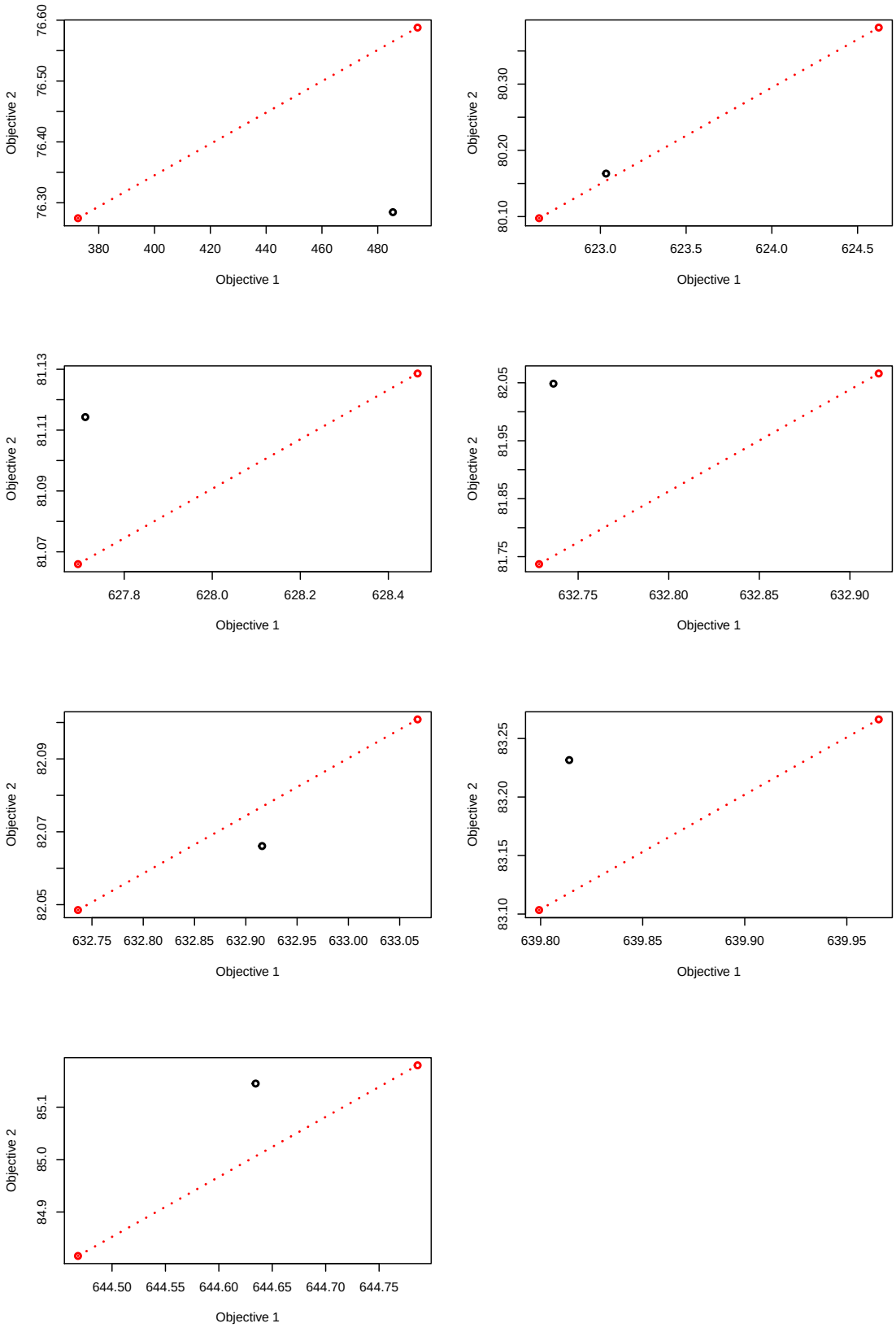


Figure 8.12: Remaining dislocations in context with the surrounding Pareto solutions

8.4.2 Scenarios response

This test involves modifying the definition of the objectives and checking whether the solutions change in the expected logical way. Two tests were performed:

- Varying price decay with volume
- De-selection of attributes

Varying price decay with volume

Three different tests were performed, the first one increasing the decay for high volumes, the second one lowering it and the third one with no decay at all, i.e. fixed price for each component independently of the volumes. The commonality index (Martin and Ishii, 1996) was calculated for all the solutions in the Pareto fronts and compared. The expected behaviour is that higher decays should result in families with more commonality, i.e. lower index. The results are plotted in figure 8.13, and they show a trend towards lower commonality indices for the solutions using high decay price curves and towards higher commonality indices for the solutions with both low decay and no decay at all.

An important point is that high decay in this context means that de price decay is more pronounced in the range of the number of units that is likely to be used. If the price curves are modelled as exponential decay, as is the case in this case study, this is not synonymous with a higher decay rate curve in general. This is visually explained in figure 8.14, the red dotted curve is a higher decay curve in general, however, when the number of units is expected to range between 2000 and 6000, the red dotted curve is actually less sensitive to volume in that range, i.e. it is a lower decay curve for all effects in this context.

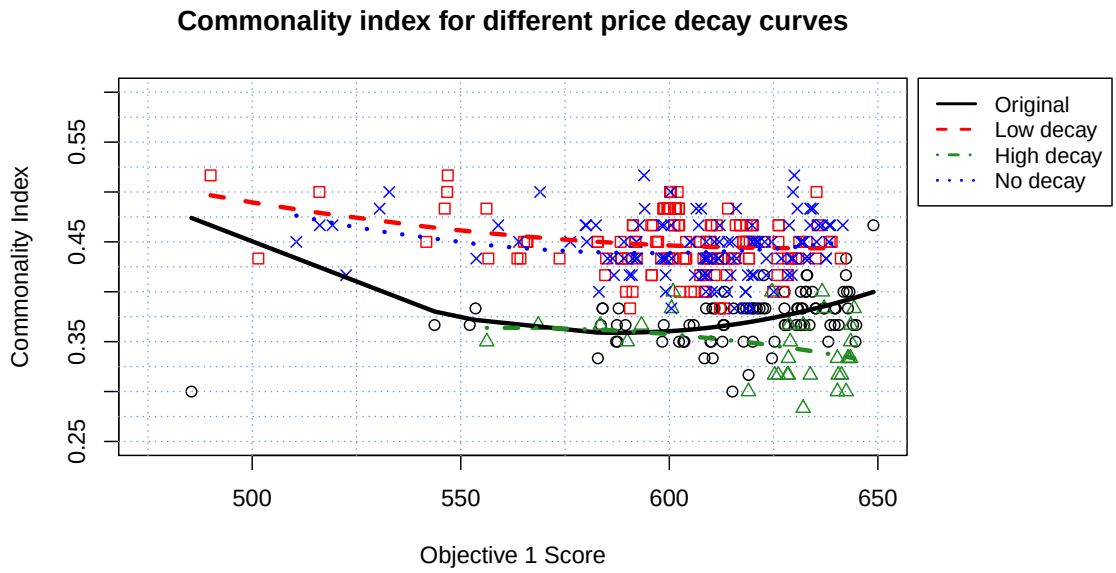


Figure 8.13: Commonality index for different price curves

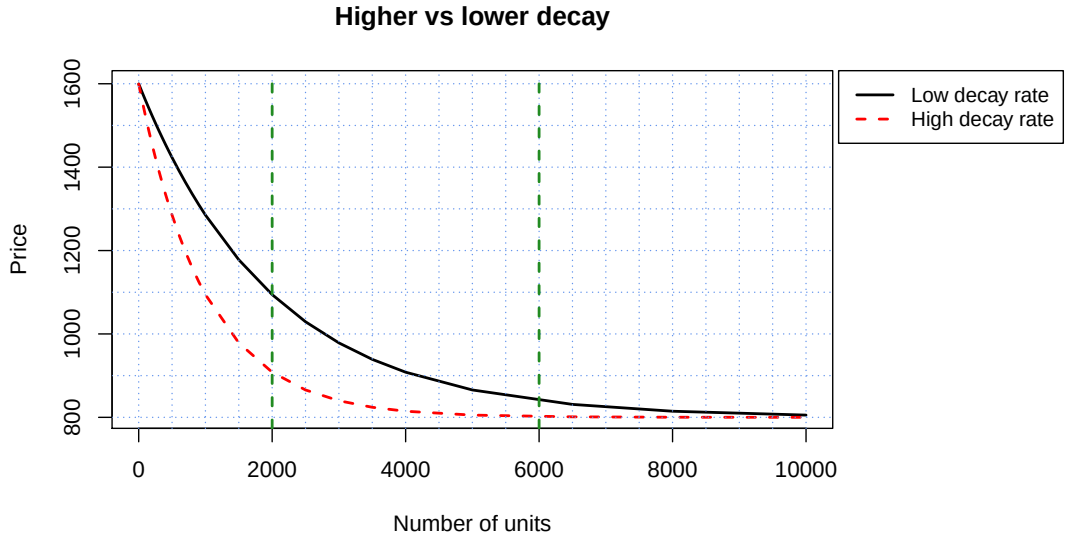


Figure 8.14: Meaning of low and high decay in this context

De-selection of attributes

This test consists on disregarding the fuel consumption as a target and calculate a new Pareto front. The expected behaviour is that the solutions should tend towards bigger engines that favour the other attributes. The new Pareto solutions under this condition were assessed against the original objectives, they are shown in red crosses in figure 8.15, and compared with the original Pareto front in black circles. It is clearly noticeable that the Pareto points found disregarding the fuel consumption do not perform as well as the original Pareto points according to the complete objectives, this is the expected behaviour.

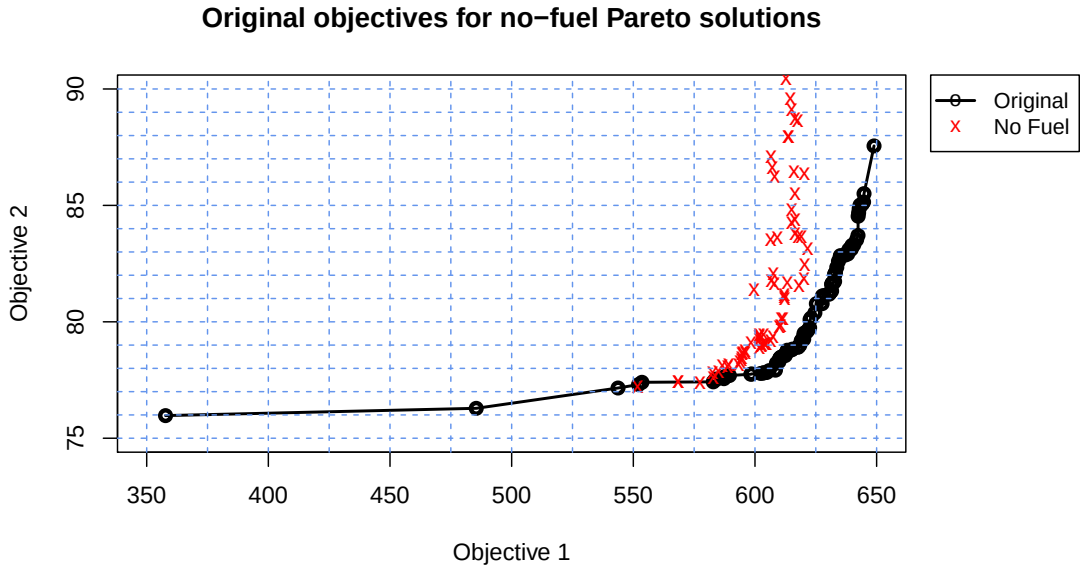


Figure 8.15: Pareto solutions with fuel deselected vs original Pareto

Then the prevalence of engines was compared to that one in the original Pareto, this is shown in figure 8.16. The x axis is divided in the different applications and each region has 6 lines, one for each engine type. The black dots show the ratio of Pareto solutions that have that engine for that application, and the red dots show the same for the Pareto solutions found disregarding fuel consumption. Observable trends include:

- Engines 5 and 6, the most powerful ones although expensive and less fuel efficient, now appear in many Pareto solutions, whereas they were unused in the original one. Engine 5 in particular starts appearing in the 4 tonnes application.

- Engine 3 is now more common at the expense of engine 2 and 4. this engine was very rare in the original Pareto due to its poor relative consumption.

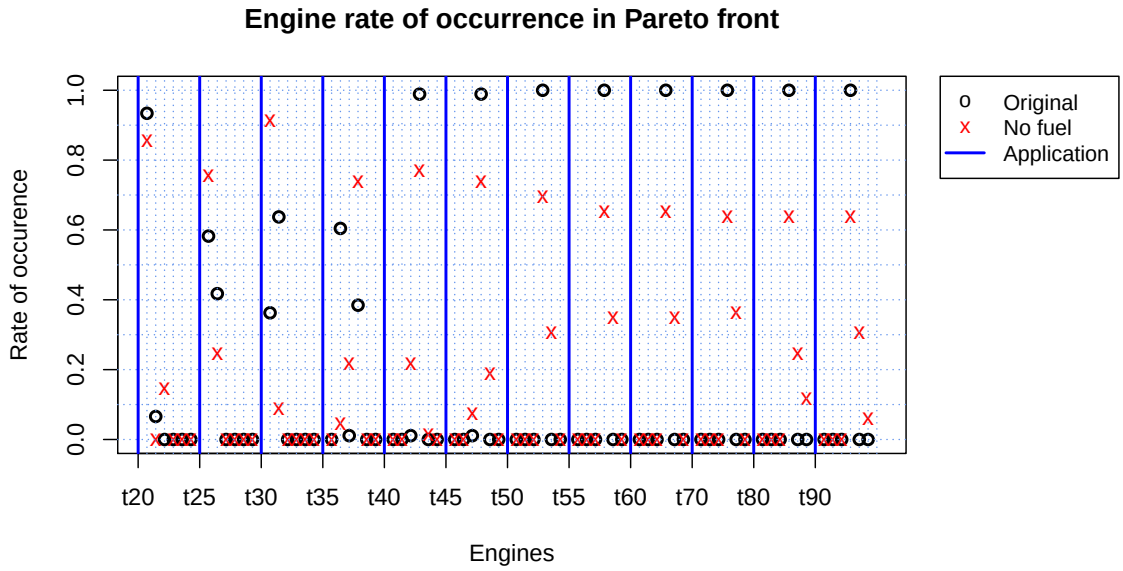


Figure 8.16: Engine prevalence original vs fuel attribute deselected

Conclusion

The variation in commonality depending on the slope of the price decay curves is clearly noticeable in the solutions and agrees with the expected behaviour. The second tests deselecting one attribute from the goodness function also agreed with the expected behaviour, hence the scenarios tests are considered as successful.

8.4.3 Matching a given family

This test consists in defining performance targets for every application and finding the family architecture that best matches those targets. For that, the fuzzy sets should be defined as very steep, and with a membership of 1 for the exact value of the target. Figure 8.17 shows an example of a fuzzy set for an application in which the fuel consumption target to match is 3.9 l/h.

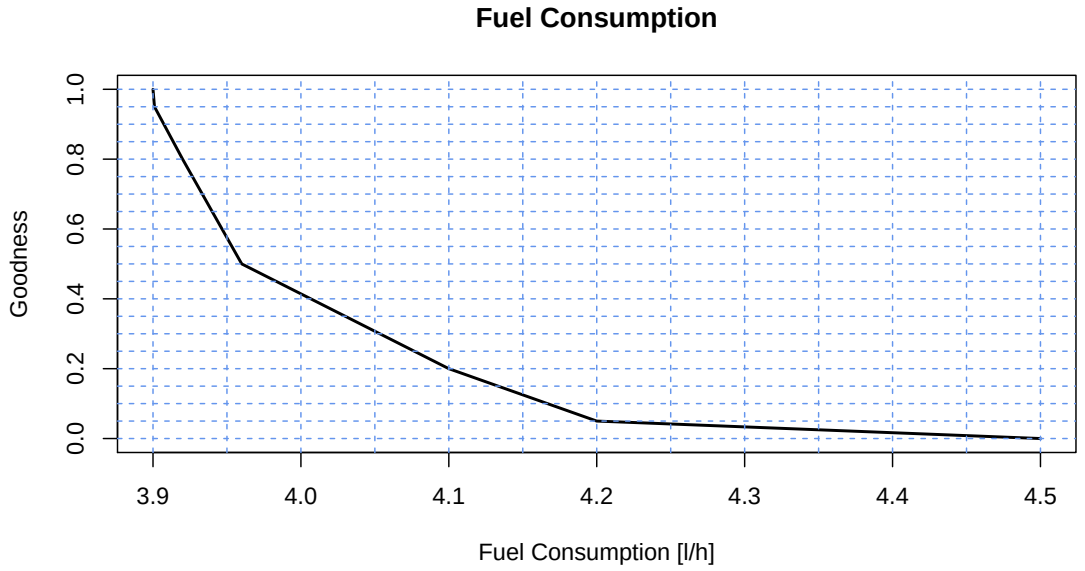


Figure 8.17: Fuzzy set to match a fuel consumption of 3.9 l/h

The targets were defined as shown in table 8.5. Then the fuzzy sets were defined in the style illustrated by figure 8.17 and the goodness for each attribute recalculated through all the possible products for each application. Two different criteria were used to select which product is the best match for each application target:

- Maximize the sum of the goodness of all the attributes
- Minimum deviation in the lowest performing attribute

Figure 8.18 shows the Pareto front, the solutions are closer to the target the more to the right they are located in the plot. This Pareto shows the trade-off between getting close to the targets and cost.

Table 8.5: Target product family

T	Fuel	SpeedU	SpeedL	LiftU	LiftL	DrawU	DrawL	GradU	GradL
2T	2.3	22	22	0.55	0.53	13000	15000	0.31	0.28
2.5T	2.5	22	22	0.55	0.53	13400	15100	0.31	0.28
3T	3.2	22	22	0.55	0.53	15300	19700	0.3	0.27
3.5T	3.4	22	22	0.55	0.53	16000	19700	0.28	0.24
4T	4.2	21	21	0.55	0.53	20500	23000	0.28	0.24
4.5T	4.4	21	21	0.55	0.53	21800	24000	0.28	0.24
5T	4.6	21	21	0.53	0.49	22200	25000	0.28	0.21
5.5T	6.7	21	21	0.53	0.49	25000	28000	0.28	0.21
6T	7.3	21	21	0.53	0.49	27000	30000	0.28	0.21
7T	7.8	21	21	0.53	0.49	29000	33000	0.28	0.21
8T	8.3	21	21	0.53	0.49	32000	35000	0.28	0.21
9T	8.8	21	21	0.53	0.49	35000	38000	0.28	0.21

The spider plots in figures 8.19 to 8.24 show the targets for each application in black solid lines and the performance of the best matches, red dashed lines for the first criterion and green dotted lines for the second criterion. It can be noticed that the achieved values for some targets of some applications are exceeded and some others are short. The former is not a problem as the objective was defined as matching the targets, without penalizing an excess. The latter means that the specified targets are not fully achievable with the available components.

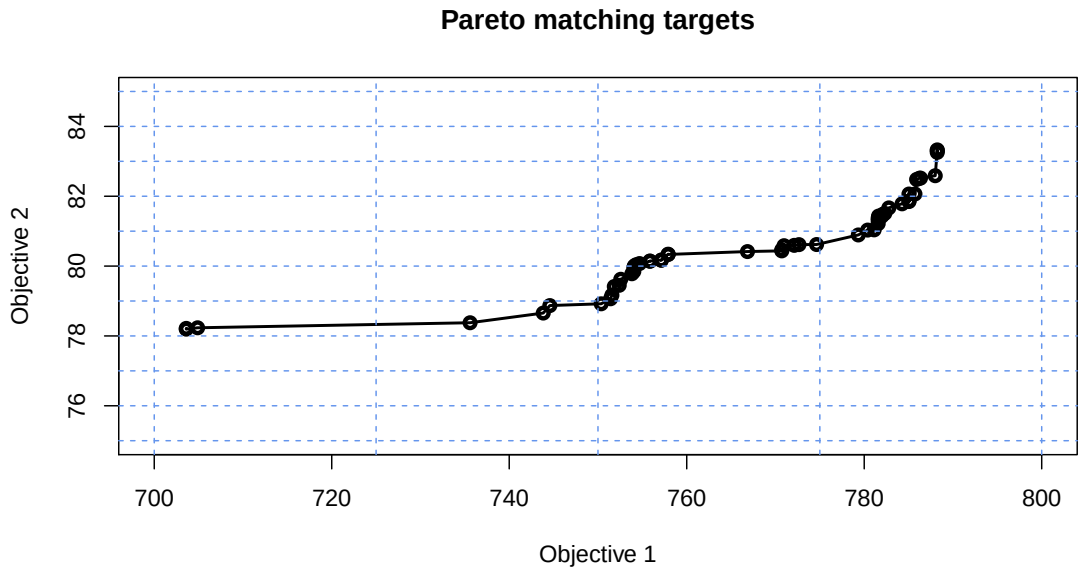


Figure 8.18: Pareto front for matching a target

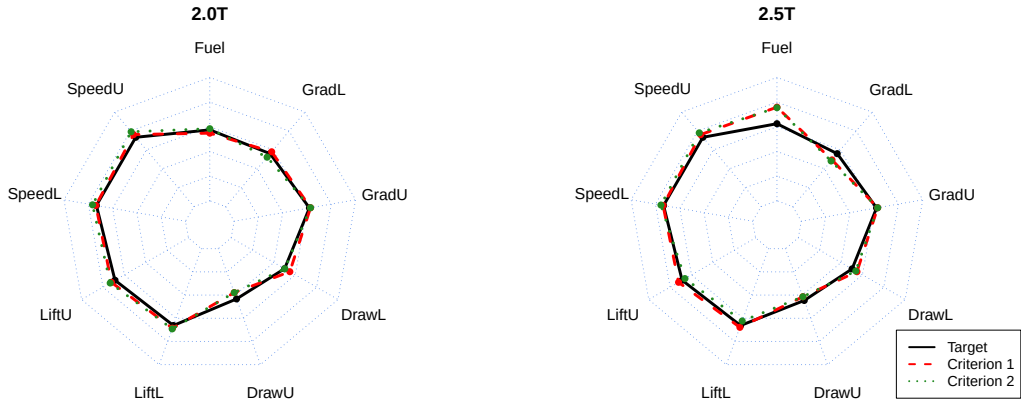


Figure 8.19: Targets vs achieved for 2T & 2.5T

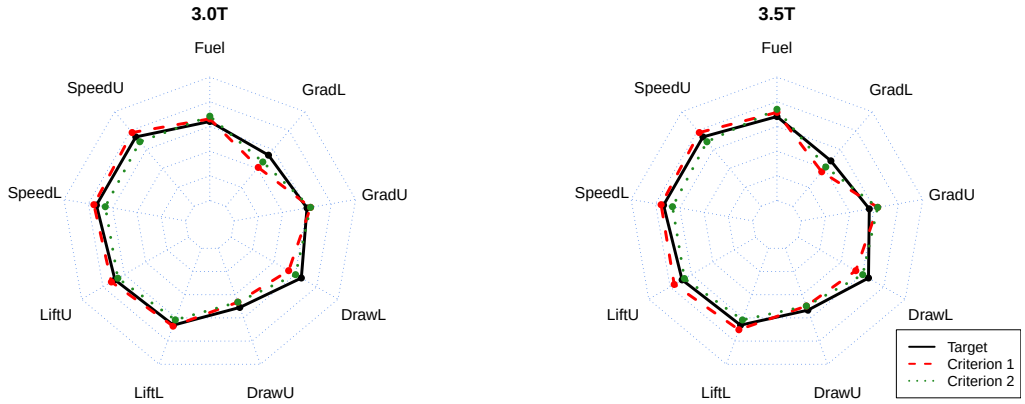


Figure 8.20: Targets vs achieved for 3T & 3.5T

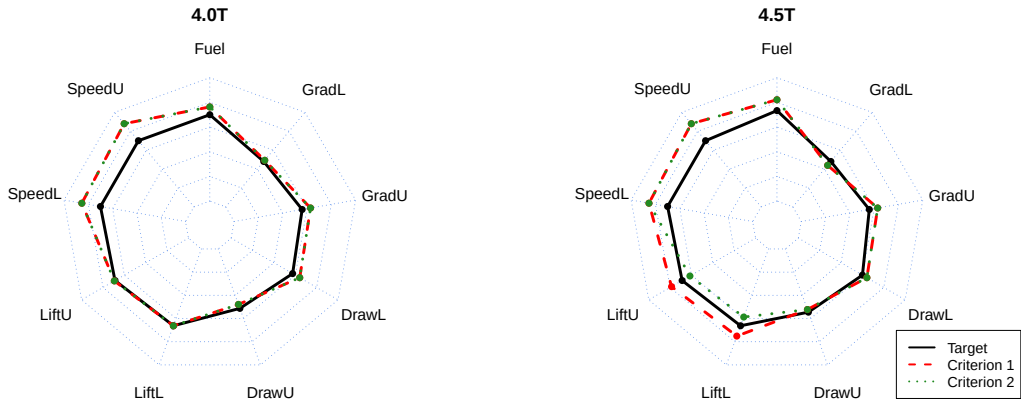


Figure 8.21: Targets vs achieved for 4T & 4.5T

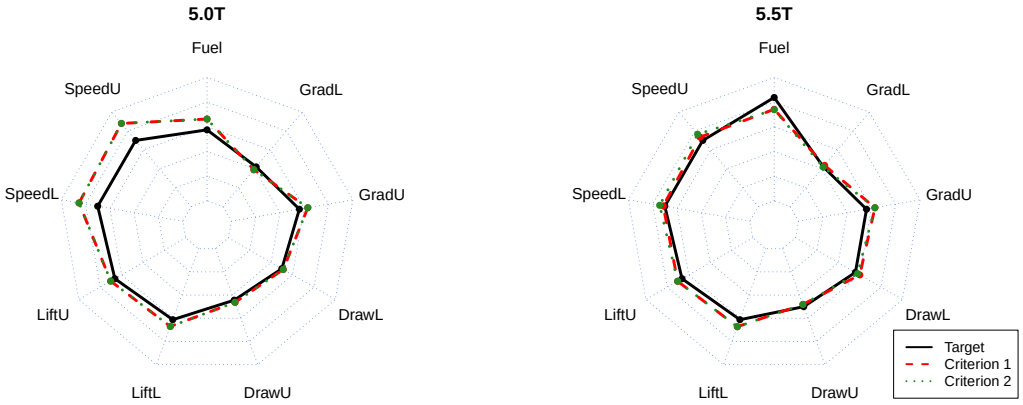


Figure 8.22: Targets vs achieved for 5T & 5.5T

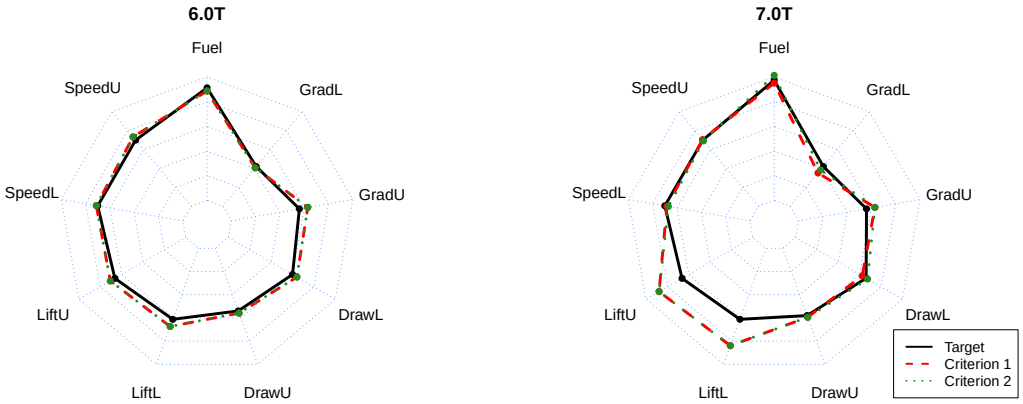


Figure 8.23: Targets vs achieved for 6T & 7T

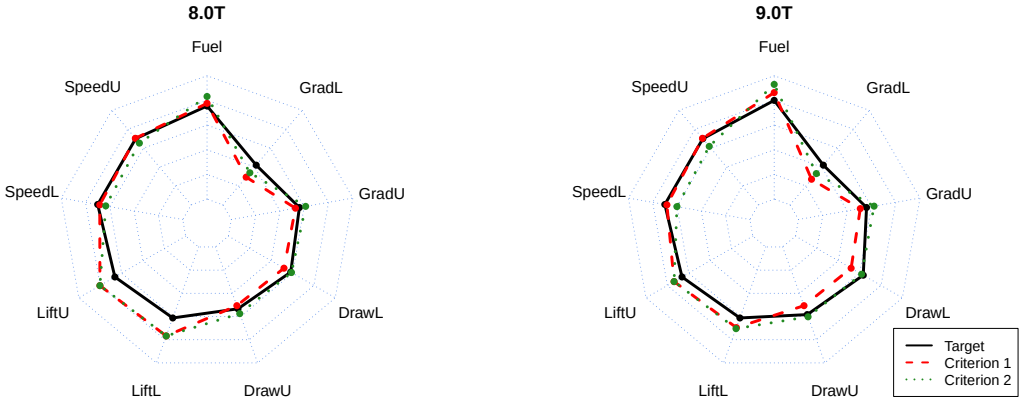


Figure 8.24: Targets vs achieved for 8T & 9T

Conclusion

This test has shown that the objective function can be defined to obtain a specifically given set of requirements for every product of the family. The result is a Pareto front that allows the visualization of the trade-off between getting close to the set of targets and reducing the costs in parts. The spider plots show the closest matches, which correspond to the rightmost point in the Pareto front.

8.5 Validation conclusion

This chapter has shown a series of successful tests that build confidence in the method described in this thesis. The tests have targeted the performance models, the method to assign an overall goodness to each product candidate, the searching algorithm, and the solution response to several changes. The validation strategy has addressed the first three parts of the validation square:

- Theoretical structural validity. The logic in the method is a direct sequencing of steps and it has been tested that it provides qualitatively appropriate solutions.
- Empirical structural validity. Characteristics of the case study have been discussed throughout the thesis and argued to make it an appropriate problem.
- Empirical performance validity. The case study and the results have been thoroughly and successfully tested.

The conclusion is that the confidence in the method is high as long as the different parts that compose it are well design and validated. The fourth part of the validation square refers to generalising method performance beyond the chosen example. This would require testing several case studies from different industrial sectors, acquiring deep knowledge of each problem, and considerable support from

experienced engineers in each field. This can not be accomplished within a PhD time frame.

Chapter 9

Evaluation, discussion and conclusions

This chapter revisits the original objectives of this thesis and the research questions and evaluates how the proposed method demonstrated over a case study in the previous four chapters provides answers to them and makes a contribution to knowledge. It also discusses some avenues for further work.

9.1 Research questions

The work presented in this thesis is intended to answer the research questions posed in chapter 1 - Introduction - section 1.4.

9.1.1 Research question 1

How can a platform strategy improve product development processes?

This is a question of a general nature and its objective is to frame the main problem and why this field is relevant. It has been addressed by a combination of three different tasks: a literature review, an analysis of the industrial practice and an exercise on a real case using the proposed method.

There is an extensive body of literature supporting the advantages of platform family design in terms of cost saving, lead times and easiness to develop a new product and a reduction of risks due to the use of well tried technologies. Those advantages come with some associated disadvantages such as possible under or overdesign for some variants, cannibalization, platform limits and unexpected failures. This combination of pros and cons makes platform family design a problem of balancing, rather than an idealised one in which the objective was as simple as building the product range with the minimum number of components.

The study of an industrial case and the feedback from expert engineers showed that the idea of platform design is well understood and rooted in the industry. There is consciousness that some form of strategy is necessary to be competitive and this fact is reflected in the product line offers, but it does not appear to be a unique or well defined way towards it. Another important point coming from the interviews is that although a model based strategy for component commonality would be welcome, the industry can be reluctant to show confidence in theoretical models without proper validation. This can potentially be a vicious circle in which the industry does not take a model due to lack of validation and the model can never be validated due to lack of taking.

The conclusion is that platform strategies is a relevant field and its adoption helps to improve the development process in many industries for all the mentioned reasons, but it also comes with downsides, what makes it a fine balanced problem.

9.1.2 Research question 2

What are the barriers to industrial adoption of product platform strategies?

The answer to this question is an analysis of the current status in the industry to find the reasons that make this problem still an ongoing area for research rather than one that has been solved and widely accepted. This has been addressed by

chapter 2 - Literature review - and chapter 4 - Industry justification -.

The industry faces several obstacles in the way of implementing platform strategies, an important one is that most existing methods look at reduced scope problems such as optimizing the size of a component so it can be shared by different products, or minimizing the number of components upon which the family is built. Other cases deal with relatively simple products in which some variables need to be scaled up and down to meet the different product requirements. While those approaches are valid, they are not addressing the problem as a whole, and require some pre-assumptions such as knowing what components are to be made common. This is an assumption necessary for any method with a priori platform identification, and as it has been discussed throughout this thesis, such approaches introduce excessive constraints that prevent some solutions from being considered. While this can be reasonable in some cases, it is not so in general, and in particular for complex machines where the effect of restricting or allowing different components can be hard to estimate. Ideally, in order to be adopted, a strategy should be flexible in that aspect.

A second important obstacle is that it is hard to balance pros and cons of the large number of possible commonality strategies across the family, and most proposed methods do not address that balance, they rather focus on minimizing the number of components or sizing some predetermined common components so that each product meets their requirements. This approach is not enough in many real cases, as the goal is not always to minimize the number of different components. A reduction of different components is typically advantageous and a means to achieve a satisfactory product family and reduce costs at the same time, but it is not the goal per se. For the industry it is important that the commonality strategy is balanced with other factors such as product performance, and a properly designed trade-off strategy is a must.

Another common feature shared by most existing methods is the two stage approach. This is an extension of a priori platform identification in which not only the common parts are selected first, they are also designed or sized without taking into account the variant parts. This approach reduces the problem computational complexity in the way of drastically reducing the number of combinations that can form a product family. However, this approach can also restrict the quality of the solutions as it involves taking some decisions without the full view that single stage methods allow, i.e. those in which all components are sized or selected at the same time. Methods in which the common components are neither identified nor chosen before the variants are more comprehensive and can offer potentially better solutions that would be obscured when any of those tasks is done beforehand, but at the expense of increased problem and computational complexity for solving it.

As mentioned in the answer to the research question 1, and perhaps the most important barrier for any method to be adopted, is that the industry requires any model based platform strategy to be somehow validated before it stands a chance of being actually used. Without validation, any method can be dismissed as simply theoretical or an idealization, and it will be hard for the decision makers to consider their results over their educated solutions, or even dedicate resources to try them.

Summarizing, these are the barriers to industrial adoption of platform design methods that have been identified in this thesis:

- Method validation and confidence.
- Applicability to real world cases.
- Methods designed for simplified problems with many assumptions.
- Pros and cons balancing.
- Two stage approaches require taking a priori decisions without a view of the whole picture.

- Single stage approaches are computationally expensive and hard to manage.

9.1.3 Research question 3

Can an alternative product platform strategy be devised that addresses these barriers?

This question is the most important one of the four, and the answer to it is the main contribution claimed in this thesis. The proposed method has been explained in chapter 5 - Developing a method to design product family architecture - and shown over a case study in chapters 6, 7 and 8.

The method has been developed over a case that is more general and has a wider scope than the typical cases found in the literature. Industrial trucks are complex machines whose behaviour depend on the components chosen in complicated ways and cannot be measured against a simple performance indicator. There are several components to select and they are involved in different ways in the different attributes under consideration. Some of the components are discrete, i.e. a choice among a number of existing options, whereas others are quasi-continuous variables such as the cylinders. Also, there is no platform architecture chosen a priori, the algorithm has absolute freedom to choose what components will be made common to what product variants, and even what products will exist, as the number of variants is an output of the algorithm itself rather than an input as it is normally the case. The proposed method generates an unspecified number of product variants to cover a predefined set of applications using a pool of components, existing or hypothetical, to build the product variants. This is a novel approach.

Other identified barrier is the way to define the objective of the platform strategy, with commonly used objectives such as minimum number of different components and the like. The method described in this thesis does not have any commonality objective, the target is to cover the predefined applications and represent the trade-off between the desirability of the solution for each application and the costs

associated with that product family. The desirability of each product for each application has been assessed by selecting the most important performance attributes for each application, calculating the figures that each design would achieve through physics based modelling and simulation, and converting each figure to a degree of goodness for each performance attribute for each application using fuzzy logic. At the same time, the costs associated to each potential product family were also calculated taking into account the relation between the cost of each component and the number of components used.

A strongly argued point in this thesis is the need to compromise between considering each single objective independently and loosing detail and definition by aggregating them into a scalar function. It is hard to define a single function that truly incorporates several objectives and reflects the preference; and the spirit of this thesis is to try and maintain the objectives as independent to generate solutions and let the decision maker articulate their preferences a posteriori. However, it is also a well known fact in optimization that keeping many objectives results in Pareto fronts almost as large as the searching space itself, and this effectively provides the decision maker with an excess of information and a lack of criteria to discern interesting from non-interesting solutions. Hence the need for compromise that this thesis has addressed by suggesting keeping the number of final objectives to two or a maximum of three, and ensure that they are grouped by conflicting interests, i.e. one of the final objectives can aggregate all those initial objectives that when they are improved, it is likely to make worse those aggregated in the second final objective. In the case study, those final objectives have been chosen as a combination of performance attributes for all the products as objective one and the cost of the components for objective two. The aggregation of performance to define the objective one has been done incorporating information given by engineering and marketing professionals in an attempt to capture their preferences.

This compromise is necessary for the method to be usable, but it is also important to point out that the decision maker can 'zoom in' any solution to observe all the individual objectives in detail.

One of the characteristics of the proposed method is that it is a single stage method, i.e. both platform and variant specific components are selected at the same time. This keeps the search open and make sure that no a priori decisions can obscure potentially good solutions. This approach was deliberate as it is superior in terms of its capability to theoretically find the best solutions, they are always a superset of the solutions found by two stage methods. The problem with single stage methods is that they are more complex to implement and time consuming as they have to look through a much bigger searching space. This difficulty has been addressed with three key points:

- Reduction of the searching space by eliminating unrealistic combinations. Due to the combinatorial nature, this point can reduce the searching space by several orders of magnitude.

- Design of an efficient searching algorithm and find appropriate parameters. This requires trial and error, this thesis has proposed a genetic algorithm but this is not a prescription.

- Improved workflow. By running simulations for all the possible combinations and storing the results in tables, the searching algorithm can run much faster, as the same simulation is never run in a redundant manner.

The final barrier, but possibly the most important one, is that the industry requires some form of validation before embracing a new method. As it was explained in chapter 3 - Methodology -, this is not a problem with a solution that can be tested in a lab or proved mathematically like a theorem. The validation approach followed in this thesis consisted on looking at every step and design tests to validate

their output. Those tests were designed in collaboration with the industry where appropriate, and their aim was to build confidence in the method. More detail on the validation exercise is provided as the answer to the research question 4.

9.1.4 Research question 4

How well could this new strategy perform in a real world industrial context?

This research question is addressed in chapters 6, 7 and 8 through a case study to test the proposed method.

The solution to the case study has shown a Pareto set of 88 product families, 44 of them are composed of 11 distinct products, 18 of 10 distinct products and 33 of 12 products. Those solutions with 10 or 11 products do not show a common clustering, e.g. an 11 products solution can have the same product to cover 2 and 2.5 tonnes while other 11 solution can have different trucks for those applications and the same truck for 3 and 3.5 tonnes. This is a result that can only be achieved thanks to the novel idea of designing a family to cover a number of applications instead of a number of products, i.e. the product clustering and the platform architecture is performed at the same time.

The problem that arises with this approach is the granularity of the applications to be considered. In the case study, the applications have been considered by the load being handled, which is a relatively high level granularity. Staying within the same case, the application called 2.5 tonnes could have been further divided according to the ratio between lifting and driving of their typical working cycles. Taken to the extreme, there may be as many applications as products will be sold, i.e. thousands of them, but increasing this number also increases the problem complexity as each application requires an assessment of what performance attributes are relevant and a quantification. This assessment involves discussions between several people and takes a considerable amount of time. It may not be feasible to assess hundreds of

applications, and hence a balance must be found between openness and feasibility with an appropriate choice of applications.

The second decision that is often, but not always, taken a priori is deciding which components will be made common to all or some of the products in the family. The method proposed does not apply any restriction in that aspect, and the solutions in the case study show many different ways in which the common components are distributed across the family. Any method with a priori platform identification - and that includes all two stage methods found in the literature - would be unable to find the variety of solutions present in the Pareto front for this case study.

The fact that the case study has considered two objectives, one for performance and one for costs, made the higher levels of commonality to appear mainly in the expensive components, such as an engine, and not so much in relatively cheap components such as cylinders, where the cost savings due to increasing commonality are minimal and the reduction in performance can be noticeable. Designers are likely to consider this point as obvious and probably would have focused their efforts on sharing the expensive parts anyway, but it must be highlighted that using the proposed method this behaviour appears as a result of the search, and not of any a priori decision. The intention of this thesis is not to debunk typical a priori decisions, but to arrive at solutions without assuming those typical decisions and hence restricting the searching space.

The method to define the target specifications with fuzzy sets allows for separating and ordering two possible families that would have otherwise checked all the boxes of a traditional requirements list. The use of fuzzy sets also allows for comparing different attributes of different products as they are measured in the same units, i.e. how good they are according to a customized definition. Hence, different products of a family can also have completely different choices of performance attributes and still be considered in the same objective function. Alternative methods such

as measuring the percentage deviation from the ideal also allow to include different kinds of attributes in the same fitness function, but they do not take into account that a deviation of 10% is not necessarily twice as bad as a deviation of 5%, it may be much worse, or unacceptable. This non-linearity is important for real industry cases.

The case study has been validated with a series of tests to increase the confidence in the results and the process that lead to them. Those steps were:

- Performance model validation: comparison between models of existing trucks and test figures. It has also been argued and proved that model accuracy is critical, as significant errors would lead to solutions that are not what they may appear. Thanks to the improved workflow in which all simulations are done in a pre-process rather than on demand as part of the fitness function during the execution of the genetic algorithm itself, the importance of the computational time to run the performance models is relatively low, since there is in general not an unmanageable number of product candidates that need to be simulated, and this process can benefit from parallelism. It is worth including all the necessary complexity in the models to ensure a sufficient level of accuracy. For example, in the case study there were 11430 product candidates to simulate, even if each simulation took 10 minutes, which is a relatively long time, it would still be possible to run all of them in 4 days with a 20 thread parallel computing process, and this part of the process only needs to be run once, independently of all the analyses and exercises that can be done later on with the remainder of the process. If, on the other hand, the process had followed the original workflow, the computation time for the simulation models would become critical, and using long run models might become infeasible, with the subsequent effect on the confidence in the solutions.

- Objectives validation: in the case study, it has been tested that the fuzzy

performance evaluation method was capable of matching the preferences of final products in a meaningful way and it is an integral part of the definition of one of the final objectives. What was not possible to prove is the match between the final aggregated objective for a whole family, as it was not possible for a human decision maker to clearly rank entire families. It is important that objective aggregation is reliable up to the point where it is feasible to be tested, as this increases the confidence that the first non-testable step can be relied upon.

- Searching algorithm validation: Statistical tests and re-runs showed Pareto convergence and a very low probability of the real Pareto front being significantly better than that one found by the algorithm.
- Dislocations: A test designed to find obvious irregularities in the solutions. Those irregularities were analysed and their correction incorporated to the algorithm as a post-process.
- Scenarios response: The algorithm was run against modified objectives and the variation of results agreed with the logical expectations.
- Matching a given family: The algorithm was tested to match a family with well defined performance requirements for each application and successfully returned a Pareto front with products close to the required figures.

As a conclusion to answer question 4, the case study has shown that the proposed method can be applied in a real industrial context.

9.2 Additional discussion

This section presents three extra features of the method that are not a direct answer to the research questions but important nevertheless.

- The products required for each application do not need to share a similar architecture, each product candidate will have its own model with its own performance attributes and goodness sets.

- When some products are already in production, they can be set as fixed and the remainder of the family built around them. An example was done and described in chapter 7 - Implementation and Results -, section 7.4. A product was defined with an engine that does not appear in the original solutions, and it was checked that this fact affected the new solutions to the point that now they also included that particular engine for other products, not only the existing one.

- If some attributes from a product in a selected region of the Pareto front do not look good, the fuzzy sets can be revised and the search rerun. This method is iterative, not necessarily gives the ideal solution at the first attempt. The performance models, however, do not need to be re-run again unless additional parts are incorporated. This is a considerable advantage and a consequence of the improved workflow introduced in chapter 6 - Case Study - section 6.6.1 in which the performance models are run before the optimization process rather than during that process.

9.3 Contribution to knowledge

This thesis has analysed a real industrial case and proposed a method to find optimal commonality strategies to design a product family considering:

- A computationally feasible multi-objective single stage method that includes potentially good solutions that could remain unexplored with two stage approaches.

- The method can be applied for product families designed from scratch, top-down, as well as for new extensions based on existing products, bottom-up.
- The output of the method is a Pareto front of - normally - two or three aggregated objectives that allows for exploring their trade-off. However, the individual performance attributes of each product are not kept hidden, and every solution can be retrieved and open to observe all of its performance attributes for a better analysis.
- The method can be used iteratively by looking at the results and revising the fuzzy sets that assess the performance attributes.
- A sub-method based on fuzzy assessment of performance to combine different attributes and provide a meaningful set of solutions.
- Use of detailed dynamic simulations to assess all the product candidates and a working flow in which this is performed as a pre-process before running the searching algorithm, improving the overall efficiency. The simulations are tailored for each product variant, and in combination with the fuzzy assessment of performance they allow for including products with different architectures.
- A series of steps to build confidence in the method to make it attractive for the industry.

9.4 Limitations and further work

The main limitations of the method, or points where care must be exercised include the sensitivity to inaccuracies in the models that calculate the different performance attributes and the residual degree of arbitrariness inherent to constructing the fuzzy sets and the equations to combine the different objectives. Those two tasks are done before the optimization and analysis of solutions, and it is important to dedicate

the necessary effort to make them accurate as the quality and usefulness of the final results depend on them.

Also, during the development of this project, some issues appeared to have a clear potential for further research. The next subsections discuss some of those issues.

9.4.1 Balancing the number of applications

This thesis has referred to an application as any intended use for a machine that can be differentiated from other uses. This is a loose definition and it is not possible to give hard rules to define where an application ends and another begins. The proposed method starts with the set of such applications that needs to be covered, and the case study has taken a predetermined set based on the author's experience with that particular type of machines. It has also been stated that the number of applications considered affects the problem complexity and solvability, so there is a balance to be found for each problem. This thesis has not addressed this issue and cannot recommend any specific method to assess that balance, this is an interesting problem on its own for further research.

9.4.2 Incorporate additional factors in the objectives

The case study has included a second objective named outgoing that includes the associated costs that need to be minimized. The costs included were those regarding the prices of the components considered in the problem, which are very dependent on the degree of commonality of the family. Apart from those, there are more costs involved that are harder to quantify, such as development time, storage, simplicity of aftersales service, etc. As it is difficult to estimate those costs accurately, unlike the costs considered in this case study which are relatively easy to measure, it is likely to be convenient to build them into a third objective.

9.4.3 Relate overall goodness with estimated selling prices

The case study has used a definition for the first objective that assumes the price for which a product can be sold is proportional to the overall goodness of that product as defined by the fuzzy sets, and then considers the overall impact on the total sales. This definition is arbitrary, as there is no evidence to prove that linear relation between the overall goodness and the selling price of the product, although some relation certainly exists. An iterative process can potentially be followed to adjust the fuzzification and/or the aggregation method so the relation between their results and the estimated selling price reflects the truth.

9.5 Conclusion

The conclusion of this thesis is that the method presented can provide a set of solutions on which the decision maker can articulate their trade-off preferences having a clear picture of the families among which they are making the selection. This can be done before any real development work is carried out and will provide the engineers with a direction of how the products can achieve the required level of performance while keeping the costs within the limits given by the selection.

The main strength of the method is the adaptability to different cases, the capability for incorporating any kind of objective that can be measured, and the development over a complex and real industrial case.

References

Agard, B. and Barajas, M., 2012. The Use of Fuzzy Logic in Product Family Development: Literature Review and Opportunities. Journal of intelligent manufacturing 23(5), pp.1445–1462.

Alizon, F., Khadke, K., Thevenot, H.J., Gershenson, J.K., Marion, T.J., Shooter, S.B. and Simpson, T.W., 2007. Frameworks for product family design and development. Concurrent Engineering, 15(2), pp.187-199.

Aljorephani, S.K. and ElMaraghy, H.A., 2016. Impact of product platform and market demand on manufacturing system performance and production cost. Procedia CIRP, 52, pp.74-79.

Arrow, K.J., 1950. A difficulty in the concept of social welfare. The Journal of Political Economy, pp.328-346

Blessing, L.T. and Chakrabarti, A., 2009. DRM: A design reseach methodology (pp. 13-42). Springer London.

Cameron, B.G., Crawley, E.F., 2014. Crafting Platform Strategy Based on Anticipated Benefits and Costs, in: Advances in Product Family and Product Platform Design: Methods and Applications. Springer.

Charnes, A. and Cooper, W.W., 1977. Goal Programming and Multiple Objective Optimizations: Part 1. European Journal of Operational Research, 1(1), pp.39-54.

Chowdhury, S., Messac, A., Khire, R.A., 2011. Comprehensive Product Platform Planning (CP3) Framework. Journal of Mechanical Design 133(10).

Collopy, P., 2009. Aerospace System Value Models: A Survey and Observations., in: AIAA 2009-6560. Presented at the AIAA SPACE 2009 Conference & Exposition, AIAA Paper 2009-6560. American Institute of Aeronautics and Astronautics, Reston, VA,, Pasadena, Ca.

Collopy, P., Hollingsworth, P., 2009. Value-Driven Design, in: AIAA 2009-7099. Presented at the 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), Hilton Head, South Carolina.

Commonality, in Cambridge dictionary, https://dictionary.cambridge.org/dictionary/english/commonality. Retrieved on 22/04/2019

Corl, M.J., Parsons, M.G., Kokkolaras, M., 2014. Optimal Commonality Decisions in Multiple Ship Classes, in: Advances in Product Family and Product Platform Design. Springer, pp. 625–645.

Da Silveira, G., Borenstein, D. and Fogliatto, F.S., 2001. Mass customization: Literature review and research directions. International journal of production economics, 72(1), pp.1-13.

Dai, Z., Scott, M.J., 2006. Effective Product Family Design Using Preference Aggregation. Journal of Mechanical Design 128(4), 659–667.

Das, I. and Dennis, J.E., 1998. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM Journal on Optimization, 8(3), pp.631-657.

Deb, K., 2001. Multi objective optimization using evolutionary algorithms (pp. 124-124). John Wiley and Sons.

Deb, K. and Sundar, J., 2006. Reference point based multi-objective optimization using evolutionary algorithms. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (pp. 635-642). ACM.

Department of Defense, 1998. DoD Modeling and Simulation (M&S) Glossary. DoD 5000.59-M

Eckert, C. M.; Stacey, M. K. and Clarkson, P. J., 2003. "The spiral of applied research: A methodological view on integrated design research." Proceedings of the 14th International Conference on Engineering Design (ICED'03), 19-21 August 2003, Stockholm, Sweden.

Eckert, C., Isaksson, O. and Earl, C., 2012. Product Property Margins: An underlying critical problem of engineering design.

Eckert, C., Earl, C., Lebjioui, S., and Isaksson, O., 2013. Components margins through the product lifecycle. In: Product Lifecycle Management for Society, 5-7 July, 2013, Nantes, Springer, pp. 39–47

Eckert, C., Albers, A., Bursac, N., Chen, H.X., Clarkson, J., Gericke, K., Gladysz, B., Maier, J., Rachenkova, G., Shapiro, D. and Wynn, D., 2015. Integrated product and process models: towards an integrated framework and review.

Ehrgott, M. and Wiecek, M.M., 2005. Mutiobjective programming. In Multiple criteria decision analysis: State of the art surveys (pp. 667-708). Springer, New York, NY.

Eischstetter, M., Muller, S., Zimmermann, M., 2015. Product Family Design With Solution Spaces. Journal of Mechanical Design 137, 121401.

Ekel, P., Kokshenev, I., Parreiras, R., Pedrycz, W. and Pereira Jr, J., 2016. Multiobjective and multiattribute decision making in a fuzzy environment and their power engineering applications. Information Sciences, 361, pp.100-119.

ElMaraghy, H. and AlGeddawy, T., 2014. Multidisciplinary domains association in product family design. In Advances in Product Family and Product Platform Design (pp. 71-89). Springer, New York, NY.

Frigg, R. and Hartmann, S., 2012. Models in Science, The Stanford Encyclopedia of Philosophy (Summer 2018 Edition), Edward N. Zalta (ed.). https://plato.stanford.edu/archives/sum2018/entries/models-science/. Retrieved on 20/03/2019

Gonzalez-Zugasti, J.P., Otto, K., Baker, J., 2000. A Method for Architecting Product Platforms. Research in engineering design 12(2), 61–72.

Hartmann, S., 1996. The world as a process. In Modelling and simulation in the social sciences from the philosophy of science point of view (pp. 77-100). Springer, Dordrecht.

Hazelrigg, G.A., 1998. A framework for Decision-Based Engineering Design. Journal of Mechanical Design 120(4), 653–658.

Hettenhausen, J., Lewis, A., Randall, M. and Kipouros, T., 2013, June. Interactive multi-objective particle swarm optimisation using decision space interaction. In 2013 IEEE Congress on Evolutionary Computation (pp. 3411-3418). IEEE.

Hirshburg, K. and Siddique, Z., 2014. A proactive scaling platform design method using modularity for product variations. In Advances in Product Family and Product Platform Design (pp. 201-219). Springer, New York, NY.

Hirshburg, K. and Siddique, Z., 2014. Designing a Lawn and Landscape Blower Family Using Proactive Platform Design Approach. In Advances in Product Family and Product Platform Design (pp. 753-776). Springer, New York, NY.

Hölttä-Otto, K., Chiriac, N., Lysy, D. and Suh, E.S., 2014. Architectural decomposition: the role of granularity and decomposition viewpoint. In Advances in Product Family and Product Platform Design (pp. 221-243). Springer, New York, NY.

Humphreys, P., 2004. Extending ourselves: Computational science, empiricism, and scientific method. Oxford University Press.

Ishibuchi, H., Tsukamoto, N. and Nojima, Y., 2008, June. Evolutionary many-objective optimization: A short review. In Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on (pp. 2419-2426). IEEE.

ISO 3691-1:2012 Industrial trucks – Safety requirements and verification – Part 1 Self propelled industrial trucks, other than driverless trucks, variable-reach trucks and burden-carrier trucks.

Jiao, J., Tseng, M.M., 2000. Fundamentals of Product Family Architecture. Integrated Manufacturing Systems 11(7), 469–483.

Jiao, J.R., Simpson, T.W. and Siddique, Z., 2007. Product family design and platform-based product development: a state-of-the-art review. Journal of intelligent Manufacturing, 18(1), pp.5-29.

Kalligeros, K., Weck, O. de, Neufville, R. de, Luckins, A., 2006. Plarform Identification Using Design Structure Matrices, in: 16th Annual International Symposium of the International Council On Systems Engineering. Presented at the 16th Annual International Symposium of the International Council On Systems Engineering.

Kasprzak, E.M., Lewis, K.E., 2000. An Approach to Facilitate Decisions Tradeoffs in Pareto Solutions Sets. Journal of Engineering Valuation and Cost Analysis 3(1), 173-187.

Khajavirad, A., Michalek, J.J. and Simpson, T.W., 2014. Solving the joint product platform selection and product family design problem: An efficient decomposed multiobjective genetic algorithm with generalized commonality. In Advances in Product Family and Product Platform Design, 271-294. Springer New York.

Khare, V., Yao, X. and Deb, K., 2003, April. Performance scaling of multi-objective evolutionary algorithms. In International conference on evolutionary multi-criterion

optimization (pp. 376-390). Springer, Berlin, Heidelberg.

Khire, R., Messac, A., 2008. Selection-Integrated Optimization (SIO) Methodology for Optimal Design of Adaptive Systems. ASME Journal of Mechanical Design 130(10), 101401.

Kim, K. and Chhajed, D., 2000. Commonality in product design: Cost saving, valuation change and cannibalization. European Journal of Operational Research, 125(3), pp.602-621.

Kim, S. and Moon, S.K., 2017. Sustainable platform identification for product family design. Journal of cleaner production, 143, pp.567-581.

Kleijnen, J.P., 2017. Regression and Kriging metamodels with their experimental designs in simulation: a review. European Journal of Operational Research, 256(1), pp.1-16.

Koski, J., 1981. Multicriterion Optimization in Structural Design. Tampere University of Technoogy (Finland).

Lampon, J.F., Cabanelas, P. and Gonzalez-Benito, J., 2017. The impact of modular platforms on automobile manufacturing networks. Production Planning & Control, 28(4), pp.335-348.

Lebeau, K., Lebeau, P., Macharis, C. and Van Mierlo, J., 2013, November. How expensive are electric vehicles? A total cost of ownership analysis. In 2013 World Electric Vehicle Symposium and Exhibition (EVS27) (pp. 1-12). IEEE.

Lebjioui, S., 2018. Investigating and Managing Design Margins throughtout the Product Development Process. Open University Thesis.

Lee, B.D., Binder, W.R., Paredis, C.J., 2014. A Systematic Method for Specifying Effective Value Models. Presented at the CSER 2014 - Conference on Systems Engineering Research. Redondo Beach, CA.

Levandowski, C., Raudberget, D.S. and Johannesson, H., 2014. Set-Based Concurrent Engineering for Early Phases in Platform Development. In ISPE CE (pp. 564-576).

Levandowski, C.E., Jiao, J.R. and Johannesson, H., 2015. A two-stage model of adaptable product platform for engineering-to-order configuration design. Journal of Engineering Design, 26(7-9), pp.220-235.

Li, Z., Pehlken, A., Qian, H. and Hong, Z., 2016. A systematic adaptable platform architecture design methodology for early product development. Journal of Engineering Design, 27(1-3), pp.93-117.

Maier, J.F., 2016. Granularity of Models in Engineering Design. Cambridge University Thesis.

Marler, R.T., Arora, J.S., 2004. Survey of Multi-objective Optimization Methods for Engineering. Structural Multidisciplinary Optimization 26(6), 369–395.

Martin, M.W., Ishii, K., 1996. Design for Variety: A Methodology for Understanding the Costs of Product Prolifiration. Presented at the Proceedings of the 1996 ASME Design Engineering Technical Conferences, ASME Irvine, CA, DTM-1610, Irvine, Ca.

Martin, M.V. and Ishii, K., 2002. Design for variety: developing standardized and modularized product platform architectures. Research in engineering design, 13(4), pp.213-235.

Mathworks, ode3 solver, https://uk.mathworks.com/help/simulink/gui/ solver.html#bq9mi4f-1 Retrieved on 17/08/2018

Messac, A., 1996. Physical Programming: Effective Optimization for Computational Design. AIAA journal, 34(1), pp. 149-158

Messac, A., Martinez, M.P. and Simpson, T.W., 2002. Effective product family design using physical programming. Engineering Optimization, 34(3), pp.245-261

Messac, A. and Mattson, C.A., 2002. Generating well-distributed sets of Pareto points for engineering design using physical programming. Optimization and Engineering, 3(4), pp.431-450.

Messac, A., Ismail-Yahaya, A. and Mattson, C.A., 2003. The normalized normal constraint method for generating the Pareto frontier. Structural and multidisciplinary optimization, 25(2), pp.86-98.

Meyer, M.H. and Lehnerd, A.P., 1997. The power of product platforms. Simon and Schuster.

Miller, S.W., Simpson, T.W., Yukish, M.A., Stump, G., Mesmer, B.L., Tibor, E.B., Bloebaum, C.L., Winer, E.H., 2014. Towards a Value-Driven Design Approach for Complex Engineered Systems Using Trade Space Exploration Tools. Presented at

the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Buffalo, New York.

Modern materials Handling, http://www.mmh.com/article/top_20_lift_truck _suppliers_2013, Retrieved on 09-10-2013.

Moon, S.K., Park, K.J. and Simpson, T.W., 2014. Platform design variable identification for a product family using multi-objective particle swarm optimization. Research in Engineering Design, 25(2), pp.95-108.

Murtagh, M., 2015. Development and Validation of a Forklift Truck Powertrain Simulation. Queens University Belfast, thesis.

Nayak, R.U., Chen, W., Simpson, T.W., 2002. A Variation-Based Method for Product Family Design. Engineering Optimization 34(1), 65–81.

Nelson, S.A., Parkinson, M.B., Papalambros, P.Y., 2001. Multicriteria optimization in product platform design. Journal of Mechanical Design 123(2), 199–204.

Otto, K., Hölttä-Otto, K., Simpson, T.W., Krause, D., Ripperda, S. and Ki Moon, S., 2016. Global views on modular design research: linking alternative methods to support modular product family concept development. Journal of Mechanical Design, 138(7).

Pander, J., 2012. Neues Konstruktionssytem Bei VW: Gleich Ist Gut. Der Spiegel, http://www.spiegel.de/auto/aktuell/neues-konstruktionssystem-bei-vw-gleich- ist-geil-a-814246.html , Google translation, retrieved on 22/03/2019

Park, J., Simpson, T.W., 2005. Development of a Production Cost Estimation Framework to Support Product Family Design. International Journal of Production Research 43(4), 731–772.

Pedersen, K., Emblemsvag, J., Bailey, R., Allen, J.K. and Mistree, F., 2000, September. Validating design methods and research: the validation square. In ASME Design Engineering Technical Conferences (pp. 1-12).

Pirmoradi, Z., Wang, G., 2011. Recent Advancements in Product Family Design and Platform-Based Product Development: A Literature Review. Presented at the Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2011, Washington DC.

Pirmoradi, Z., Wang, G., Simpson, T.W., 2014. A Review of Recent Literature in Product Family Design and Platform-Based Product Development, in: Advances in Product Family and Product Platform Design: Methods and Applications. Springer.

Purshouse, R.C., Deb, K., Mansor, M.M., Mostaghim, S. and Wang, R., 2014, July. A review of hybrid evolutionary multiple criteria decision making methods. In 2014 IEEE congress on evolutionary computation (CEC) (pp. 1147-1154). IEEE.

Raudberget, D., Levandowski, C., Isaksson, O., Kipouros, T., Johannesson, H. and Clarkson, J., 2015. Modelling and assessing platform architectures in pre-embodiment phases through set-based evaluation and change propagation. Journal of Aerospace Operations, 3(3, 4), pp.203-221.

Read, D., 2004. Utility theory from Jeremy Bentham to Daniel Kahneman.

Ripperda, S. and Krause, D., 2017. Cost effects of modular product family structures: Methods and quantification of impacts to support decision making. Journal of Mechanical Design, 139(2), p.021103.

Robertson, D., Ulrich, K., 1998. Planning for Product Platforms. Sloan Management review 19–31.

Sangiovanni-Vincentelli, A. and Martin, G., 2001. Platform-based design and software design methodology for embedded systems. IEEE Design & Test of Computers, 18(6), pp.23-33.

Scott, M.J. and Antonsson, E.K., 1999. Arrow's theorem and engineering design decision making. Research in Engineering Design, 11(4), pp.218-228.

Shu, E.S., 2005. Flexible Product Platforms. MIT Thesis.

Siddique, Z., Rosen, D.W., Wang, N., 1998. On the Applicability of Product Variety Design Concepts to Automotive Platform Commonality, in: Proceedings of the 1998 ASME Design Engineering Technical Conferences, ASME, Atlanta, GA, DTM-5661. Atlanta, GA.

Simon, H.A., 1956. Rational choice and the structure of the environment. Psychological review, 63(2), p.129.

Simpson, T.W., Maier, J.R. and Mistree, F., 2001. Product platform design: method and application. Research in engineering Design, 13(1), pp.2-22.

Simpson, T.W., D'Souza, B.S., 2004. Assessing Variable Levels of Platform Commonality Within a Product Family Using a Multiobjective Genetic Algorithm. Concurrent Engineering - Research and Applications 12(2), 119–129.

Simpson, T.W., 2004. Product platform design and customization: Status and promise. Ai Edam, 18(1), pp.3-20.

Simpson, T.W., Jiao, R.J., Siddique, Z. and Hölttä-Otto, K., 2014. Product family and product platform design: looking forward. Advances in Product Family and Product Platform Design, pp.777-787.

Six Sigma Institute. https://www.sixsigma-institute.org/Six_Sigma_DMAIC _Process _Measure_Phase_Process_Capability.php, retrieved on 25/02/2019

Sobek II, D.K., Ward, A.C. and Liker, J.K., 1999. Toyota's principles of set-based concurrent engineering. MIT Sloan Management Review, 40(2), p.67.

Song, B., Luo, J. and Wood, K., 2019. Data-Driven Platform Design: Patent Data and Function Network Analysis. Journal of Mechanical Design, 141(2), p.021101.

Szumska, E., Jurecki, R. and Pawelczyk, M., 2019. Assessment of Total Costs of Ownership for Midsize Passenger Cars with Conventional and Alternative Drive Trains. Communications-Scientific letters of the University of Zilina, 21(3), pp.21-27.

Teegavarapu, S., Summers, J. and Mocko, G. 2008. Case Study Method for Design Research: A Justification. Proceedings of the ASME Design Engineering Technical Conference.

Thevenot, H., Simpson, T.W., 2007. A Comprehensive Metric for Evaluating Component Commonality in a Product Family. Journal of Mechanical Design 18(6), 577–598.

Thiele, L., Miettinen, K., Korhonen, P.J. and Molina, J., 2009. A preference-based evolutionary algorithm for multi-objective optimization. Evolutionary computation, 17(3), pp.411-436.

Ulrich, K., 1995. The Role of Product Architecture in the Manufacturing Firm. Research Policy 24(3), 419–440.

Ulrich, K., Eppinger, n.d. Product design and development, 2012, 5th ed. McGraw Hill.

VDI 2198, 2012. Type Sheet for Industrial Trucks. Verein Deutscher Ingenieure

von Neumann, J., Morgenstern, O., 1944. Theory of Games and Economic Behaviour. Full text available from Archive.org

Wang, H., Olhofer, M. and Jin, Y., 2017. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. Complex & Intelligent Systems, 3(4), pp.233-245.

Watanabe, N., 1993. Statistical Methods for Estimating Membership Functions. Japanese Journal of Fuzzy Theory and Systems 5(4), 589-601

Weck, O. de, Suh, E.S., Chang, D., 2003. Product Family and Platform Portfolio Optimization, in: DETC'03. Presented at the 2003 ASME Design Engineering Technical Conferences, Chicago, IL.

Wierman, M., 2010. An Introduction to the Mathematics of Uncertainty. Creighton University.

Wilson, Q. "Classic American Cars" DK Publishing, 1997.

Wolpert, D.H. and Macready, W.G., 1997. No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), pp.67-82.

Yin, R.K., 2017. Case study research and applications: Design and methods. Sage publications.

Zadeh, L.A., 1965. Fuzzy Sets. Information and Control 8(3), 338–353.

Zadeh, L.A., 2001. A New Direction in AI: Toward a Computational Theory of Perception. AI Magazine 22(1), 73.

Zainal, Z., 2007. Case study as a research method. Jurnal Kemanusiaan, 5(1).

Zapico, M., Eckert, C., Jowers, I. and Earl, C., 2015. Towards Product Platform Introduction: Optimising Commonality of Components. In DS 80-7 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 7: Product Modularisation, Product Architecture, systems Engineering, Product Service Systems, Milan, Italy, 27-30.07. 15 (pp. 023-032).

Appendix - R code

This appendix includes the code used for the main algorithm in the case study. The databases used are not included. Some data has been obfuscated due to confidentiality issues.

```
# CODE FOR THE MAIN ALGORITHM AND NECESSARY FUNCTIONS
# SOME DATA VECTORS HAVE BEEN OBFUSCATED FOR CONFIDENTIALITY ISSUES

rm(list=ls())

## OVERALL GOODNESS FUNCTION ##################################################
# Function to calculate the overall goodness of a product candidate for a particular
# application
# 'product' is a vector with the 9 goodness attributes of a product candidate.
# 'application' is the application, used only if the weights are different for each

Overall_goodness = function(Product, Application){

  Weights = c(10,6,6,4,5,2,2,1,1)
  s = 1

  a = Product ^ s;
  top = sum(a * Weights);
  bottom = sum(Weights);

  g = (top/bottom) ^ (1/s)

  return(g)
}

###############################################################################

##  OBJECTIVES FUNCTION ######################################################
# Function to calculate the objectives of a family candidate
# The input is a 12 element vector with the position of each product in the
# corresponding application set

Objectives = function(Family){

  # eFamily is a 12x6 matrix where each row is a product and the columns are
  # the five components and overall goodness
  eFamily = matrix(0,12,6)
  for (v in 1:12){
    eFamily[v,] = eval(parse(text=aNames[v]))[Family[v],]
  }

  # Incoming
  Max_number_of_products = c(//OBFUSCATED 12 ELEMENTS VECTOR//)
  # Number of products to be built for each application.
  Max_sell_prices = c(//OBFUSCATED 12 ELEMENT VECTOR//)
  #Maximum price for individual value = 1

  # Outgoing  Eng 6, pump 9, tyre 5, gear 7, cylinder 14
  # Create column matrices to store the amount of each component used
  Engines = matrix(0,6,1)
  Pumps = matrix(0,9,1)
  Tyres = matrix(0,5,1)
  Gears = matrix(0,7,1)
  Cylinders = matrix(0,14,1)

  # Add the components of each product multiplied by the target production units
  for (z in 1:12 ){
    Engines[eFamily[z,1],1] = Engines[eFamily[z,1],1] + Max_number_of_products[z]
    Pumps[eFamily[z,2],1] = Pumps[eFamily[z,2],1] + Max_number_of_products[z]
    Tyres[eFamily[z,3],1] = Tyres[eFamily[z,3],1] + Max_number_of_products[z]
    Gears[eFamily[z,4],1] = Gears[eFamily[z,4],1] + Max_number_of_products[z]
    Cylinders[eFamily[z,5],1] = Cylinders[eFamily[z,5],1] + Max_number_of_products[z]
  }
  rm(z)

  # Minimum prices for each component assuming an infinite number of units
  Base_prices = matrix(c(//OBFUSCATED 70 ELEMENTS VECTOR//),14,5)

  # Max prices are the amount to add to the base prices when buying a single unit.
  # It is not necessarily equal to base prices as in this example
  Max_prices = Base_prices

  # Coefficients for the exponential decay function relating number of units
  # with price per unit
  Decay_coefficients = 0.0005 * matrix(c(//OBFUSCATED 70 ELEMENTS VECTOR//),14,5)

  Cost = matrix(0,5,1)   # Total cost of components

  # Go through each component, calculate their cost and add them all
  for (w in 1:6){
    price = Base_prices[w,1] +
      Max_prices[w,1] * exp(-Decay_coefficients[w,1] * Engines[w,1])
    Cost[1,1] = Cost[1,1] + price * Engines[w,1]
  }
  rm(w, price)
```

```
  for (w in 1:9){
    price = Base_prices[w,2] +
      Max_prices[w,2] * exp(-Decay_coefficients[w,2] * Pumps[w,1])
    Cost[2,1] = Cost[2,1] + price * Pumps[w,1]
  }
  rm(w, price)
  for (w in 1:5){
    price = Base_prices[w,3] +
      Max_prices[w,3] * exp(-Decay_coefficients[w,3] * Tyres[w,1])
    Cost[3,1] = Cost[3,1] + price * Tyres[w,1]
  }
  rm(w, price)
  for (w in 1:7){
    price = Base_prices[w,4] +
      Max_prices[w,4] * exp(-Decay_coefficients[w,4] * Gears[w,1])
    Cost[4,1] = Cost[4,1] + price * Gears[w,1]
  }
  rm(w, price)
  for (w in 1:14){
    price = Base_prices[w,5] +
      Max_prices[w,5] * exp(-Decay_coefficients[w,5] * Cylinders[w,1])
    Cost[5,1] = Cost[5,1] + price * Cylinders[w,1]
  }
  rm(w, price)

  # Objective 1
  Incoming = sum(eFamily[,6] * Max_number_of_products * Max_sell_prices)

  # Objective 2
  Total_cost = sum(Cost)

  # Objectives are divided by 1e6 to keep them in more manageable numbers

  Ob1 = Incoming/1e6
  Ob2 = Total_cost/1e6
  Ob = c(Ob1, Ob2)
  return(Ob)
}

## END OF FUNCTIONS DECLARATION


## MAIN ALGORITHM ##################################################################

# Load performance data
setwd('D:/Location/Csvs')
p20 = read.csv('T20.csv')
p25 = read.csv('T25.csv')
p30 = read.csv('T30.csv')
p35 = read.csv('T35.csv')
p40 = read.csv('T40.csv')
p45 = read.csv('T45.csv')
p50 = read.csv('T50.csv')
p55 = read.csv('T55.csv')
p60 = read.csv('T60.csv')
p70 = read.csv('T70.csv')
p80 = read.csv('T80.csv')
p90 = read.csv('T90.csv')

pNames = c('p20', 'p25', 'p30', 'p35', 'p40', 'p45', 'p50', 'p55', 'p60', 'p70',
           'p80', 'p90')
gNames = c('g20', 'g25', 'g30', 'g35', 'g40', 'g45', 'g50', 'g55', 'g60', 'g70',
           'g80', 'g90')
aNames = c('a20', 'a25', 'a30', 'a35', 'a40', 'a45', 'a50', 'a55', 'a60', 'a70',
           'a80', 'a90')

# pNames have the physical performance of the different attributes
# gNames have the goodness of the diferent performance attributes
# aNames have the components and overall goodness. This are the sets passed to
# the searching algorithm

# Fuzzy level cuts
Fvalues = c(0,0.05,0.2,0.5,0.8,0.95,1)

# Read the data for each performance attribute of each application corresponding
# to each level cut
Fsetsdata = read.csv('Fsets1.csv')
Fsetsdata = as.matrix(Fsetsdata)

# Re-arrange Fdatasets into a 9x7x12 array.
# 1st index = performance attribute
# 2nd index = level cut
# 3rd index = application
Fsets = array(0,dim=c(9,7,12))
```

```
for (i in 1:12){
  Pos = 10*(i-1)+1 # Positions 1, 11, 21, 31, ...
  Fsets[,,i] = Fsetsdata[Pos:(Pos+8),1:7] # Database for each application
}
rm(Fsetsdata, Pos, i)

# Eval each of the 12 performance sets,
for (i in 1:12){
  # Eval one application performance set
  pInst = eval(parse(text = pNames[i]))
  pInst = as.matrix(pInst)
  # One application goodness set
  gInst = matrix(0, nrow(pInst), ncol(pInst))
  colnames(gInst) = colnames(pInst)
  # Copy the component info
  gInst[,1:5] = pInst[,1:5]
  # Loop for all possible products of the application
  for (j in 1:length(pInst[,1])){
    # Loop for all performance features of that possible product
    for (k in 1:9){
      # Interpolate the performance in the two closest goodness level cuts
      gInst[j,(k+5)] = approx(Fsets[k,,i],Fvalues, pInst[j,(k+5)], rule=2)$y
    }
    rm(k)
  }
  # Assign this goodness matrix to the corresponding application matrix
  assign(gNames[i], gInst)
  rm(pInst, gInst, j)
}
rm(i)
rm(p20, p25, p30, p35, p40, p45, p50, p55, p60, p70, p80, p90)

# Loop to delete products for which at least one attribute is zero
for (i in 1:12){
  gThis = eval(parse(text = gNames[i]))
  for (j in length(gThis[,1]):1){
    if (prod(gThis[j,6:14]) == 0){
      gThis = gThis[-j,]
    }
  }
  assign(gNames[i], gThis)
  rm(gThis,j)
}
rm(i)

# Call the overall goodness function for each possible product of each
# application
for (i in 1:12){
  gNow = eval(parse(text = gNames[i]))
  aNow = gNow[,1:5]
  aNow = cbind(aNow,0)
  colnames(aNow) = c(colnames(gNow)[1:5], 'OverallGoodness')
  for (j in 1:length(aNow[,1])){
    aNow[j,6] = Overall_goodness(gNow[j,6:14])
  }
  rm(j)
  assign(aNames[i], aNow)
  rm(aNow, gNow)
}
rm(i)
rm(g20, g25, g30, g35, g40, g45, g50, g55, g60, g70, g80, g90)
rm(Fsets, Fvalues, gNames, pNames) # Clean workspace

# Workspace ready, start algorithm
################################################################################
################################################################################

# Genetic algorithm parameters
# Number of individuals in each generation
sPopulation = 60
# Parameter to choose how many of the new gen come from crossing parents
uParents = 40
# Paramenter to choose how many come from mutating existing parents
Mutation = 10
# Parameter to choose the probability of mutation for each component in the
# mutated offspring
Mutation_chance = 0.15
# Number of iterations to run
nIterations = 100000
# Number of historical Pareto front to record
nRecord = 50

# Prepare initial data
# Size of each set of possible product to choose a random row
Sizes = c(nrow(a20), nrow(a25), nrow(a30), nrow(a35), nrow(a40), nrow(a45),
```

```
        nrow(a50), nrow(a55), nrow(a60), nrow(a70), nrow(a80), nrow(a90))

# Initialize pareto front with 1 random vector
Pareto = matrix(0,1,14)
colnames(Pareto) = c('t20', 't25', 't30', 't35', 't40', 't45', 't50', 't55', 't60',
                     't70', 't80', 't90', 'Obj1', 'Obj2')
First = ceiling(Sizes * runif(12))
Pareto[1,1:12] = First
Pareto[1,13:14] = Objectives(First)
# Size of the Pareto front
sPareto = 1
rm(First)

# Create first generation
Population = matrix(0,sPopulation,12)
for (i in 1:sPopulation){
  Population[i,] = ceiling(Sizes * runif(12))
}
rm(i)
Population = cbind(Population,matrix(0,sPopulation,2))

# Settings to record Pareto fronts at diferent points in history
# Geometric progression: nRecord elements from 1 to nIterations
k = nIterations ^ (1/(nRecord - 1))
Milestones = k^(seq(0,(nRecord-1)))
Milestones = ceiling(Milestones)


Milestones[nRecord] = min(Milestones[nRecord], nIterations)
# To avoid a numerical issue in which the final milestone was one
# unit greater than nRecord
rm(k)

# Prevent two repeated milestones
for (i in 2:nRecord){
  if (Milestones[i] <= Milestones[i-1]){
    Milestones[i] = Milestones[i-1] + 1
  }
}
rm(i)


######################################################################
######################################################################
##### ALTERNATIVE LOGARITHMIC MILESTONES #############################
#ln = log(nIterations)
#temp = ln/nRecord
#Milestones =seq(1,nRecord) * temp
#Milestones = ceiling(exp(Milestones))
#rm(ln, temp)
#Milestones[nRecord] = min(Milestones[nRecord], nIterations)
# To avoid a numerical issue in which the final milestone was one
# unit greater than nRecord
######################################################################
######################################################################

# Initial Pareto plus each recorded iteration
Pareto_historical = matrix(NA,(nRecord),2)
Pareto_historical[1,1:2] = Pareto[1,13:14]
fill = 2 # To start filling the Pareto historical in the second row

############### LOOP BEGINS HERE #################################################
tic = Sys.time()

for (iter in 1:nIterations){
  # Run objectives function for each individual in the current population
  for (i in 1:sPopulation){
    Population[i,13:14] = Objectives(Population[i,1:12])
  }
  rm(i)

  # Add elements of this generation to Pareto front
  # Create matrix for the first non-dominated front
  # Create matrix for the points not belonging in the non-dominated front
  # 15th column will be a weigth: 1 for nondominated1, 0.8 for nondom2,
  # 0.6 for nondom3 and 0.4 for nondom4
  Non_dominated1 = matrix(0,sPopulation,15)
  Non_dominated1[,15] = 1
  Remainder = matrix(0,sPopulation,14)
  nondom1 = 1 # Variable representing the row of Non_dominated1 to be filled
  rem = 1 # Variable representing the row of Remainder to be filled

  # Find 1st front

  for (i in 1:sPopulation){
```

```
  # Each individual in this generation is checked agains every other until
  # either other individual dominates it (Dominated will become 1) or the
  # while loop reaches the end of the generation. Dominated is set as 0 at
  # the beginning
  Dominated = 0
  j = 1
  while (Dominated == 0 && j <= sPopulation){
    if (j != i){
      if (Population[i,13] < Population[j,13] &&
          Population[i,14] > Population[j,14]){
        Dominated = 1
        # Condition for point i to be dominated by j breaks the loop
      }
    }
    j = j + 1
  }
  # Once the loop is broken, if the point is not dominated, add
  # to Non_dominated1
  if (Dominated == 0){
    Non_dominated1[nondom1,1:14] = Population[i,]
    nondom1 = nondom1 + 1
  }
  else { # Else, add the point to Remainder
    Remainder[rem,] = Population[i,]
    rem = rem + 1
  }
}
rm(i,j)
# Delete rows of Non_dominated1 and Remaninder not filled
Non_dominated1 = Non_dominated1[Non_dominated1[,1] != 0, , drop = FALSE]
Remainder = Remainder[Remainder[,1] != 0, , drop = FALSE]
rm(nondom1)

# Find 2nd front

s2 = length(Remainder[,1])
Non_dominated2 = matrix(0,s2,15)
Non_dominated2[,15] = 0.8
Remainder2  = matrix(0,s2,14)

nondom2 = 1
rem = 1

if (s2 > 0){
# This if was introduced to correct a bug in which the loop still
# executed once even though there was no row in Remainder 2 to
# allocate individuals
  for (i in 1:s2){
    Dominated = 0
    j = 1
    while (Dominated == 0 && j <= s2){
      if (j != i){
        if (Remainder[i,13] < Remainder[j,13] &&
            Remainder[i,14] > Remainder[j,14]){
          Dominated = 1
        }
      }
      j = j + 1
    }
    if (Dominated == 0){
      Non_dominated2[nondom2,1:14] = Remainder[i,]
      nondom2 = nondom2 + 1
    }
    else {
      Remainder2[rem,] = Remainder[i,]
      rem = rem + 1
    }
  }
  rm(i,j,s2)
}
Non_dominated2 = Non_dominated2[Non_dominated2[,1] != 0, , drop = FALSE]
Remainder2 = Remainder2[Remainder2[,1] != 0, , drop = FALSE]
rm(Remainder, nondom2)

# Find 3rd front

s3 = length(Remainder2[,1])
Non_dominated3 = matrix(0,s3,15)
Non_dominated3[,15] = 0.6
Remainder3  = matrix(0,s3,14)

nondom3 = 1
rem = 1

if (s3 > 0){
```

```
  for (i in 1:s3){
    Dominated = 0
    j = 1
    while (Dominated == 0 && j <= s3){
      if (j != i){
        if (Remainder2[i,13] < Remainder2[j,13] &&
            Remainder2[i,14] > Remainder2[j,14]){
          Dominated = 1
        }
      }
      j = j + 1
    }
    if (Dominated == 0){
      Non_dominated3[nondom3,1:14] = Remainder2[i,]
      nondom3 = nondom3 + 1
    }
    else {
      Remainder3[rem,] = Remainder2[i,]
      rem = rem + 1
    }
  }
  rm(i,j,s3)
}
Non_dominated3 = Non_dominated3[Non_dominated3[,1] != 0, , drop = FALSE]
Remainder3 = Remainder3[Remainder3[,1] != 0, , drop = FALSE]

rm(Remainder2, nondom3)

# Find 4th front

s4 = length(Remainder3[,1])
Non_dominated4 = matrix(0,s4,15)
Non_dominated4[,15] = 0.4
Remainder4  = matrix(0,s4,14)

nondom4 = 1
rem = 1

if (s4 > 0){
  for (i in 1:s4){
    Dominated = 0
    j = 1
    while (Dominated == 0 && j <= s4){
      if (j != i){
        if (Remainder3[i,13] < Remainder3[j,13] &&
            Remainder3[i,14] > Remainder3[j,14]){
          Dominated = 1
        }
      }
      j = j + 1
    }
    if (Dominated == 0){
      Non_dominated4[nondom4,1:14] = Remainder3[i,]
      nondom4 = nondom4 + 1
    }
    else {
      Remainder4[rem,] = Remainder3[i,]
      rem = rem + 1
    }
  }
  rm(i,j,s4)
}
Non_dominated4 = Non_dominated4[Non_dominated4[,1] != 0, , drop = FALSE]
Remainder4 = Remainder4[Remainder4[,1] != 0, , drop = FALSE]

rm(Remainder3, nondom4, Dominated, rem, Remainder4)
# End of fronts search

# Update Pareto - testing points of Non_dominated1
sDom = length(Non_dominated1[,1])
sPareto_0 = sPareto # Number of elements against which the points have to
# be checked, sPareto_0 is created as a local variable for this task,
# this may be unnecessary

for (i in 1:sDom){
  # Measure point against all previous Pareto points.
  Dominated = FALSE
  Dominates = FALSE
  Redundant = FALSE

  j = 1
  while (Dominated == FALSE && Redundant == FALSE && j <= sPareto_0){
    # Dominated = TRUE breaks the loop, the point is discarded
    # Redundant = TRUE also does the same
    Dominated = (Non_dominated1[i,13] < Pareto[j,13]) &&
```

```
      (Non_dominated1[i,14] > Pareto[j,14])
    Redundant = prod(Non_dominated1[i,1:12] == Pareto[j,1:12])
    Dominates = (Non_dominated1[i,13] > Pareto[j,13]) &&
      (Non_dominated1[i,14] < Pareto[j,14])

    # if point i dominates point j
    if (Dominates == TRUE){
      Pareto = Pareto[-j,] # Remove point j from the Pareto front
      sPareto_0 = sPareto_0 - 1 # Decrease the count of the Pareto size
      sPareto = sPareto - 1 # Decrease the count of the Pareto size
    }
    else {
      j = j+1 # go to the next point j
    }
  }

  # The nested loop completed, now check if the point is dominated (it can
  # only be by the last point j, otherwise the loop would have broken before)
  if (Dominated == FALSE && Redundant == FALSE){
    # Add point i to the Pareto front
    Pareto = rbind(Pareto, Non_dominated1[i,1:14])
    # Increase the count of the Pareto size
    sPareto = sPareto + 1
  }
}
rm(i,j, Redundant, Dominates, Dominated, sDom, sPareto_0)

# Record everything
if (sum(iter == Milestones) == TRUE && iter != 1){ # True if iter is equal to
  # any element in Milestones except 1, which is already filled
  # l is the amount of columns that need to be added to Pareto historical
  l = 2 * sPareto - length(Pareto_historical[1,])
  if (l > 0){
    Pareto_historical = cbind(Pareto_historical, matrix(NA, (nRecord), l))
    }
  Paretoline = matrix(0,1,2*sPareto) # Line to be added to Pareto historical
  for (p1 in 1:sPareto){
    # Fill odd cells with objective 1
    Paretoline[1,(2*p1 - 1)] = Pareto[p1,13]
    # Fill even cells with objective 2
    Paretoline[1,(2*p1)] = Pareto[p1,14]
  }
  rm(p1)
  # Add the line to Pareto historical
  Pareto_historical[fill,1:(2*sPareto)] = Paretoline
  rm(Paretoline)
  # Increase the row to which the next historical data will be added
  fill = fill +1
}

# Create next generation here
Parents = rbind(Non_dominated1, Non_dominated2, Non_dominated3, Non_dominated4)

for (rParents in 1:length(Parents[,1])){
  Parents[rParents, 15] = Parents[rParents,15] * runif(1)
}
# Re-order Parents according to the 15th element
Parents = Parents[order(Parents[,15], decreasing = TRUE),]

# sOffspring is the mimimum of the size of parents and the setting for children
# coming directly from existing parents
sOffspring = min(uParents, length(Parents[,1]))

# Now create direct offspring
Nextgen = matrix(0,sPopulation,14)
for (offsp in 1:sOffspring){  # Crossing between parents
  # First parent is chosen sequentially second parent is chosen randomly
  Second_parent =  ceiling(length(Parents[,1])*runif(1))
  Offspring_line = matrix(0,1,12) # This child
  # Each product has a 50% chance of coming from the 1st or the 2nd parent
  for (comp in 1:12){
    rChoice = runif(1)
    if (rChoice > 0.5){
      Offspring_line[1,comp] = Parents[offsp,comp]
    }
    else {
      Offspring_line[1,comp] = Parents[Second_parent,comp]
    }
  }
  Nextgen[offsp,1:12] = Offspring_line
}
rm(offsp,comp)
# Now mutated indivuduals
for (offsp in 1:Mutation){
   Nextgen[(sOffspring+offsp),1:12] =
```

```
      Parents[(((offsp - 1) %% length(Parents[,1])) + 1),1:12]
    for (mut in 1:12){
      mProb = runif(1)
      if (mProb < Mutation_chance){
        Nextgen[(sOffspring+offsp),mut] = ceiling(Sizes[mut]*runif(1))
      }
    }
  }
  rm(offsp,mut,mProb)
  # Now fill with random families
  for (offsp in (sOffspring + Mutation + 1):sPopulation){
    Nextgen[offsp,1:12] = ceiling(Sizes * runif(12))
  }


  # Test for repeated individuals
  for (i in 1:uParents){
    for (j in (i+1):uParents){
      if (prod(Nextgen[i,1:12] == Nextgen[j,1:12]) == 1){
        for (mut in 1:12){
          mProb = runif(1)
          if (mProb < Mutation_chance){
            Nextgen[j,mut] = ceiling(Sizes[mut]*runif(1))
          }
        }
      }
    }
  }

  Population = Nextgen

  if (iter %% 5000 == 0){
    toc1 = Sys.time() - tic
    print(toc1)
  }

} # End searching loop
rm(Nextgen, Non_dominated1, Non_dominated2, Non_dominated3, Non_dominated4,
   Offspring_line, Parents, Population, fill, i, iter, j, l, mProb, mut, Mutation,
   nRecord, offsp, rChoice, rParents, Second_parent, sOffspring, sPareto, sPopulation,
   toc1)


toc = Sys.time() - tic
print(toc)
```